Fanatical Support for AWS Product Guide

Release 2018-04-19-17:04

April 19, 2018

CONTENTS

IMPORTANT: This is a PDF version of the Product Guide, and is intended to be used for point-in-time offline reference purposes only. The authoritative version of this document lives online at https://manage.rackspace.com/aws/docs and will contain the latest updates.

This Product Guide is designed to provide a detailed look at how Rackspace delivers our **Fanatical Support for AWS** offering. It covers core concepts such as the *AWS account structure* and Rackspace *service levels*, and advanced concepts such as provisioning bastion access via *Rackspace Passport* and accessing audit logs via *Rackspace Logbook*.

For general information on the offering, please visit https://www.rackspace.com/aws.

To sign up, visit https://cart.rackspace.com/aws.

CHAPTER

ONE

GETTING STARTED

It is extremely easy to get started experiencing Fanatical Support for AWS.

1.1 Create your Rackspace account

The first step is to create your Rackspace account. Visit https://cart.rackspace.com/aws and follow the instructions to establish your account.

1.2 Add a new AWS account

Once you have created your Rackspace account navigate to the Fanatical Support for AWS Control Panel. Login using the credentials you established during the signup process above.

Note that all new Rackspace accounts undergo a thorough review to minimize fraud. This process can take several minutes to several hours to complete, depending on the details of your signup. You will not be able to proceed until the verification is complete. If you would like to expedite the verification process, please *Contact Us*.

Once you are logged in you will see an option to add a new AWS account. Provide the relevant details and select a *service level*. We will immediately provision you with a new AWS account ready for your use. You can click the "Log in to AWS Console" link to go straight to AWS, though we encourage you to first review our *Recommended Network Configuration*.

If at any time you need assistance from a Racker, please do not hesitate to *Contact Us*.

1.3 Use an existing AWS account

Our general recommendation is to create a new AWS account as it will be provisioned immediately and will already include all of our best practice configuration ready for you to use. If you have an existing AWS account that you would like to use with our services, please see *Transferring Existing AWS Accounts to Rackspace* for additional details regarding the process.

CHAPTER

TWO

AWS ACCOUNTS

Each Rackspace account can house one or more AWS accounts. By default, you can create up to five new AWS accounts via the Fanatical Support for AWS Control Panel. If you need more than five accounts, please open a ticket to request a limit increase. In addition to creating new AWS accounts, you may also *transfer existing AWS accounts* to Rackspace for management.

Each AWS account provides a top-level administrative control boundary for the resources that are a part of it. While it is possible to leverage Amazon's Identity and Access Management (IAM) platform to isolate certain resource access, we typically recommend provisioning an AWS account per application deployment phase (e.g. development, staging, and production), thereby allowing you to assign different users in your organization access to one or more of the accounts without complex IAM policies. In this example, developers could be granted access to provision EC2 instances, RDS databases, etc. in your development and staging accounts, but be restricted to read access of the resources in your production account.

In addition to being a strong permission boundary, AWS accounts also provide a convenient construct for tracking expenses, since by default, both AWS and Rackspace charges are grouped by AWS account. For example, if 4 separate AWS accounts are used called app1-dev, app1-prod, app2-dev, app2-prod, it is very easy to see how much is being spent on each application environment. We highly encourage the use of tagging for more fine grained tracking of expenses within accounts, but tagging is more complicated, certain resources may be missing tags resulting in unallocated cost, and not all AWS resource types support tagging. AWS accounts provide a great default cost allocation construct.

Lastly, using separate AWS accounts per environment gives you the flexibility to select different Rackspace *service levels* for each environment, since Rackspace service levels are applied at the AWS account level. For example, you may opt for the Navigator service level on your development account while using the Aviator service level for your production environment.

As is described later in this document, several Fanatical Support for AWS features (such as *Rackspace Logbook*) are available in both cross-account and account-specific views, enabling unified visibility across multiple AWS accounts.

2.1 Account Defaults

For all AWS accounts managed by Rackspace, whether created new via the Fanatical Support for AWS Control Panel or created directly with AWS and transferred to Rackspace, we automatically apply several default settings to the account based on best practices we have developed in cooperation with AWS. You should not change or disable any of these default settings, as they are critical to our delivery of Fanatical Support.

- AWS IAM (Identity and Access Management)
  - Setup an IAM role named "Rackspace" for ongoing access to the account (see *AWS Identity and Access Management (IAM)* for additional details)
  - Set the IAM account password policy for all passwords
    - At least 12 characters in length

        - Contain at least one uppercase character
        - Contain at least one lowercase character
        - Contain at least one number
        - Contain at least one symbol
        - Not one of the previous 24 passwords used
    - Set the AWS account alias to "rax-<account_uuid>". For accounts transferred to Rackspace, the alias is only modified if a custom one does not already exist.
    - Create an IAM role named "AWSConfig" for use by the AWS Config service
    - Create an IAM role named "RackspaceTools" to allow us to provide you with *Compass*
    - Create an IAM role named "RackspaceDefaultEC2Role" along with an attached IAM policy named "RackspaceDefaultEC2Policy" which can be attached to EC2 instances to provide access to AWS Systems Manager and the CloudWatch EC2 Agent.
- AWS S3 (Simple Storage Service)
    - Create a bucket named "<account_uuid>-logs" in the US West 2 (Oregon) region
        - Enable versioning and apply an S3 bucket lifecycle policy to the "<account_uuid>-logs" bucket that expires files after 365 days and permanently removes deleted files after 90 days
        - Set an S3 bucket policy on the "<account_uuid>-logs" bucket to allow write access from CloudTrail
    - Create a bucket named "<account_uuid>-ssmoutput" in the US West 2 (Oregon) region
        - Apply an S3 bucket lifecycle policy to the "<account_uuid>-ssmoutput" bucket that deletes files after 60 days
- AWS CloudTrail
    - Configure AWS CloudTrail in each AWS region to log to the S3 bucket named "<account_uuid>-logs"
    - Configure an SNS topic named "rackspace-cloudtrail" in each region and subscribe it to a region-specific Shared Management Services SQS queue for use by the *Rackspace Logbook* service
- AWS Config
    - Configure AWS Config in each AWS region to log to the S3 bucket named "<account_uuid>-logs"
    - Configure an SNS topic named "rackspace-awsconfig" in each region and subscribe it to a region-specific Shared Management Services SQS queue for use by Rackspace tooling
- AWS SNS (Simple Notification Service)
    - Create SNS topics named "rackspace-support", "rackspace-support-standard", "rackspace-support-urgent", "rackspace-support-emergency" in each region and subscribe it to a region-specific Shared Management Services SQS queue for use by our *Rackspace Watchman* service

2.2 Transferring Existing AWS Accounts to Rackspace

While the Fanatical Support for AWS Control Panel enables the ability to easily provision new AWS accounts, there may be situations where you would like to transfer an existing AWS account to Rackspace for management. This is also supported, and once complete, will allow Rackspace management tooling and expertise to function against your existing account.

This process involves formally assigning your AWS account to Rackspace for management, which can be initiated by submitting a request via the Fanatical Support for AWS Control Panel. The following information is required:

- AWS Account Number
- Legal Company Name
- Legal Company Address
- Authorized Signatory Name (the individual who can legally give authorization to assign your AWS account to Rackspace)
- Authorized Signatory Email Address

Once your request is received, both AWS and our teams will review your account. After the review, the authorized signatory will receive a legal document from AWS via DocuSign that must be signed. From there, a few additional steps will be required of you to prep your account (you will receive a ticket with details) and then your account can be transitioned.

This process typically takes 2-4 weeks from start to finish, which is somewhat dependent on you since certain steps of the process require action on your part. Please monitor your email and the Support Tickets section of the Fanatical Support for AWS Control Panel for tickets that require your action.

Note that transferring an existing AWS account to Rackspace does not count against the limit of new AWS accounts you are able to provision via the Fanatical Support for AWS Control Panel.

2.2.1 Minimum Account Requirements

In order for an existing AWS account to be transitioned to Rackspace, it must meet our minimum account requirements, which include:

- No access keys exist for the root account
- No EC2 Classic in use
- The account is not consolidated under a payer account or serving as a payer account with linked child accounts

These requirements **must** be met before the account can be transitioned to Rackspace.

2.3 Offboarding

While we hope to serve you for life, should you ever decide that you no longer require Rackspace's management of your AWS account we can work with you to transition your account to a direct relationship with AWS. You would retain access to all AWS resources, but would lose access to Rackspace tooling such as *Logbook* and *Compass* as well as Rackspace's AWS expertise and service. If you are considering making this change, please *contact your Account Manager* for further assistance.

CHAPTER

THREE

SERVICE LEVELS

Fanatical Support for AWS combines tooling and automation with human experts to deliver a world-class experience. We offer two service levels, Navigator and Aviator, which are selected for each AWS account we support.

- Navigator: "I want to do most things myself, but I want access to Rackspace's AWS experts and tools."
- Aviator: "I want Rackspace to operate and manage my AWS environments for me or with me."

For details on what is included in each service level, including details on levels of support for each AWS service, download our `Fanatical Support for AWS Service Overview`.

3.1 Features: Tooling and Automation

A curated set of Rackspace developed and best of breed AWS ecosystem tools:

- AWS Account Generation Pre-Configured with *Rackspace Best Practices*
  - Service Levels: Navigator and Aviator
  - Features
    - AWS root account credentials encrypted and locked away
    - MFA enabled on root account and secret configuration key encrypted and locked away
    - No named IAM users; all AWS access via single, dynamically scoped IAM role and temporary STS credentials
    - CloudTrail and AWS Config enabled with centralized logging
    - Separate AWS accounts per environment (e.g. development, staging, production)
- Access to AWS Trusted Advisor
  - Service Levels: Navigator and Aviator
  - Features
    - Access to all Trusted Advisor checks
- *Rackspace Compass*
  - Service Levels: Navigator and Aviator
  - Features
    - Best Practices: More than 350 automated best practice checks evaluated against your AWS accounts
    - Cost Optimization: Billing dashboards, savings reports, cost alerting, and Reserved Instance purchase recommendations

        * Inventory Management: Cross-account and cross-region resource inventory, per service usage details, resource tagging reports, and more
        * Security: CloudTrail, Config, VPC and security group analysis, perimeter assessments, and IAM and permission reporting
        * Utilization: CPU and network heat maps and CloudWatch historical data retention and analysis
- *Rackspace Passport*
    - Service Levels: Aviator only (customer and Rackspace use)
    - Features
        * On-demand provisioning of bastions for secure network access to VPC resources
        * Automatic, temporary credential management via the In-Instance Credential Management Service
        * Full logging
- In-Instance Credential Management Service (powered by ScaleFT)
    - Service Levels: Aviator only (customer and Rackspace use)
    - Features
        * Automatic certificate authority and SSH key rotation across your fleet of EC2 instances
        * Temporary, fast expiring keys with silent renewal

3.2 Features: Human Experts

Tap into an army of certified AWS architects and engineers ready to deliver Fanatical Support to your business 24x7x365. Available via ticket and phone.

- AWS best practice and architecture consultation from 100% AWS certified experts
    - Service Levels: Navigator (standard use cases) and Aviator (customized to your specific application)
- Hands-on management and assistance for all supported AWS services
    - Service Levels: Aviator only
- EC2 operating system management
    - Service Levels: Aviator only
    - Features
        * Amazon Linux: 2015.03+, Red Hat Enterprise Linux: 6 & 7, CentOS: 6 & 7, Ubuntu LTS Versions: 14.04 & 16.04, Windows Server 2008 R2*, Windows Server 2012 R2, Windows Server 2016
        * Configuration, Optimization, Patching, Upgrades
    - Prerequisites: The following agents must be installed and working on your EC2 instances in order to be supported by Rackspace
        * Passport - The ScaleFT server agent allows Rackspace support team to access your instances via SSH (Linux) or RDP (Windows)
        * SSM - The AWS Systems Manager agent allows Rackspace to manage your EC2 instances remotely (instance configuration, maintenance of agent versions and updates, OS patching, software inventory monitoring)
- *Rackspace Watchman*

  - Service Levels: Aviator only
  - Features
    * Rackspace AWS certified engineer response to CloudWatch alarms 24x7x365
    * Set up CloudWatch alarms to a pre-configured SNS topic or let us do it for you
- Custom CloudFormation template creation
  - Service Levels: Aviator only
- Data restoration support (for EC2 and RDS exclusively)
  - Service Levels: Aviator only

* Support for Windows Server 2008 R2 is contingent on enabling an alternative means of access (beyond ScaleFT) for Rackspace to manage your instances. Please work with your Support team prior to deploying new instances running Windows Server 2008 R2.

3.3 Response Time SLAs

Rackspace will respond to your support requests submitted to us via ticket in the following timeframes. All requests should be made directly to Rackspace and we will escalate to AWS directly, if needed.

- Emergency (Business-Critical System Outage / Extreme Business Impact): If Rackspace Infrastructure monitoring and alerting services determines your AWS Services are inaccessible from the public internet, which may result in the inability to complete business transactions, our initial response to emergency monitoring alarms will occur within fifteen minutes (Aviator service level only; monitoring response is not included in the Navigator service level).
- Urgent (Production System Outage / Significant Business Impact): If your AWS Services are functioning improperly or at less than optimal performance and the failure is impacting business transactions, our initial response is 60 minutes. Customers must call Rackspace immediately after creating the Urgent ticket to trigger the one hour response guarantee. This severity is only available for the Aviator service level.
- High (Production System Impaired / Moderate Business Impact): If your AWS Services are functioning improperly or at less than optimal performance, but the failure is not impacting business transactions, our initial response to your support request submitted to us via a ticket will occur within four hours at the Aviator or Navigator service levels.
- Normal (Issues and Requests / Minimal Business Impact): If your AWS Services are functioning normally but you have a time sensitive request, question, or issue that needs addressed, our initial response to your support request submitted to us via a ticket will occur within 12 hours at the Aviator and Navigator service levels.
- Low (General Information, Questions, and Guidance): If your AWS Services are functioning normally but you require information or assistance, wish to schedule maintenance, or require the completion of any other non-immediate tasks, our initial response to your support request submitted to us via a ticket will occur within 24 hours at the Aviator and Navigator service levels.

CHAPTER

FOUR

PRICING

Your monthly service fees will be calculated by pooling the AWS infrastructure charges from all of your AWS accounts at the same service level. Learn more about monthly service fee calculations in the *Billing* section.

Service fees are charged in addition to AWS infrastructure rates. Service Fees for AWS Reserved Instances and Spot Instances are calculated based on the corresponding on-demand list rates. AWS infrastructure is charged at the list rates on the AWS website. You can view your service fees in the Fanatical Support for AWS Control Panel.

Please note that the AWS free tier is not available to Fanatical Support for AWS accounts.

CHAPTER

FIVE

AVIATOR INFRASTRUCTURE MANAGEMENT

5.1 Summary

Based on your desired infrastructure management preference, some AWS environments built under the Fanatical Support for AWS Aviator service level are managed directly via the AWS console, while others are managed through a process known as Infrastructure as Code (IaC), specifically using an AWS service named CloudFormation.

5.2 Management via the AWS console

Organizations that are not used to Infrastructure as Code (IaC) practices can benefit from a simplified management of their AWS environment via the AWS console. With this type of management, small changes can be implemented quicker. If your environment is managed via the AWS console, you have the added flexibility to make changes by yourselves, if you wish to do so. However, new resources need to be deployed by Rackspace.

5.3 Management via IaC using CloudFormation

Organizations that are comfortable with IaC practices can have their AWS environments managed using CloudFormation. With this method of management, all changes must be requested via tickets and deployed by Rackspace. Changes via the AWS console are not allowed since they can conflict with CloudFormation management, resulting in downtime, data loss, or delays to reconcile these manual changes. It is important that all changes to your environment are managed with CloudFormation.

5.4 Why use CloudFormation?

If you are comfortable to work with a strict change management process and to give up the ability to make changes via the AWS console, managing your environment using CloudFormation can provide the following benefits:

- Ability to automatically and consistently rebuild or duplicate environments
- Version-controlled and quality checked infrastructure changes
- Automated testing of the interconnected parts of an environment
- Inherent Disaster Recovery plans for your infrastructure

5.5 What resources are managed with CloudFormation?

If your environment is managed via IaC, you should consider all of it under the control of CloudFormation, unless the Rackspace Support team instructs you otherwise.

The following resources are never maintained with CloudFormation, however we still recommend engaging the Rackspace Support team for any changes!

- IAM User Console Passwords
- IAM User Access Keys
- Elastic Beanstalk Environments
- Route 53 External Zones

Resources in the list above can be modified using the AWS console without the risk of creating configuration drift. If you're unsure, please contact the Rackspace Support team.

5.6 What if I make changes outside of CloudFormation?

If your environment is managed via IaC, making changes outside of CloudFormation creates configuration drift, meaning that the code we use to maintain your environment is no longer in sync with what actually exists within your AWS account. This may result in:

- Downtime or data loss, as manual infrastructure changes are overwritten (configuration changed, resources recreated or deleted) by a CloudFormation stack update
- Delays implementing changes you request, as the CloudFormation templates and parameters must be reconciled with manual infrastructure changes before proceeding

The impact to the infrastructure may be wider than just the directly modified resources, and in some circumstances may require downtime or a rebuild of much of the environment. Even the smallest change can have wide reaching consequences!

If you have been forced to make manual infrastructure changes in an emergency, please contact the Rackspace Support team as soon as possible to document the changes that were applied.

5.7 I don't want to use CloudFormation, can I opt out?

Yes, Rackspace allows your environment to be managed via the AWS console. If you wish to discuss the strategy for ongoing management of your environments, please don't hesitate to contact your Account Manager, who will be happy to discuss the options with you!

5.8 Terraform and GitHub Support (Beta)

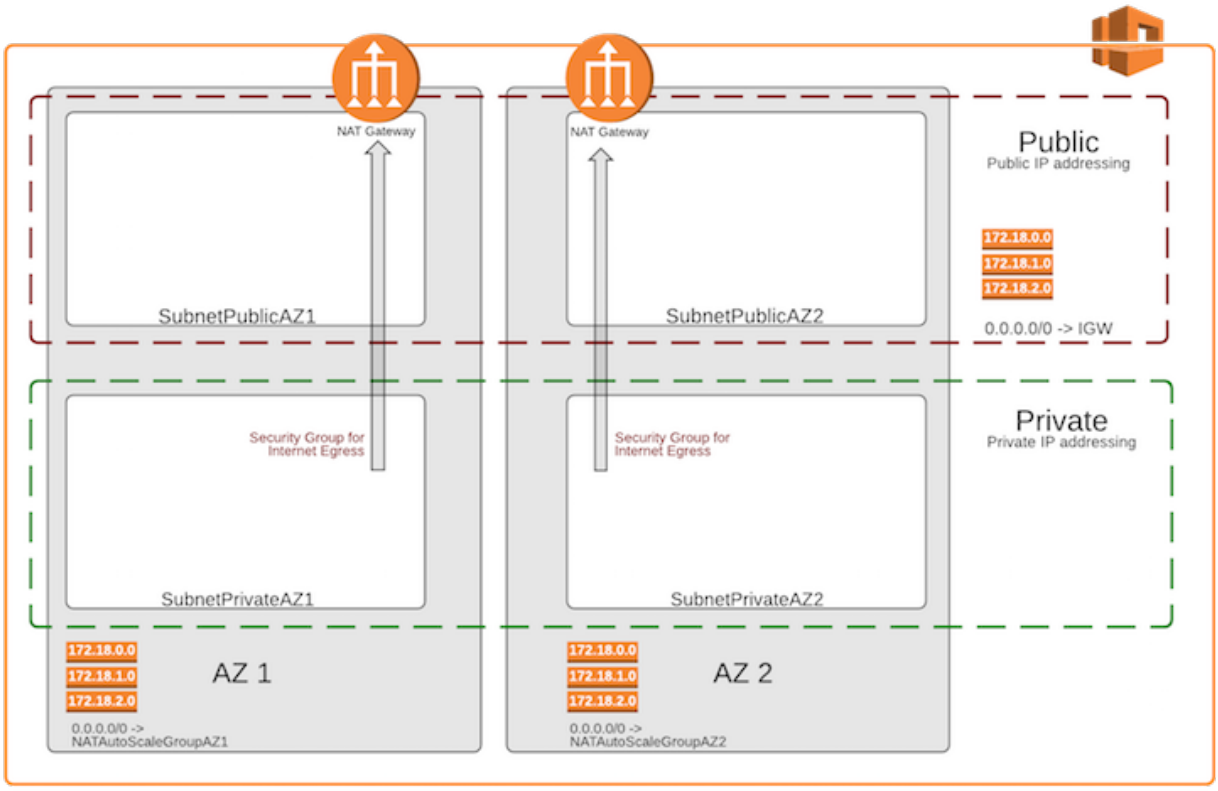
Fanatical Support for AWS is currently beta testing *support for Terraform and GitHub* as an alternative to having your infrastructure managed by CloudFormation. If you're interested in learning more about this future option, please reach out to your Account Manager.

CHAPTER

SIX

RECOMMENDED NETWORK CONFIGURATION

This section describes the necessary scaffolding and processes to create the initial AWS network environment for Rackspace customers using AWS through the Fanatical Support for AWS offering. A CloudFormation template and additional supporting scripts will be used to create the initial network and all of its necessary components, thus providing Public and Private subnets for EC2 instances and other AWS services.



This includes:

- A Single VPC
- Availability Zones (AZ) Options
  - Two AZ deployments are the standard
  - Three AZ deployment to address specific application requirements
- Subnets
  - Public Tier - could be accessible from the Internet

  - Private Tier - could access the Internet via a NAT environment
  - Subnets in each Tier will have the same network masks
- Highly Available Outbound NAT (HA-NAT) with Elastic IP - for EC2 gateways in the Private Subnets
- Security Groups - primary method to isolate and secure workloads
- Tagging - to address Rackspace billing and operational processes

You can access the template by downloading it from `here`.

Note: The template will create AWS resources for which you will be charged (for example, EC2 NAT gateways).

6.1 CloudFormation

There are two important concepts to understand when using AWS CloudFormation: *templates* and *stacks*. A template is used to describe your AWS resources and their properties. When you create a stack, AWS CloudFormation provisions the resources that are described in the template.

To learn more, view the AWS documentation on stacks and templates.

6.1.1 Rackspace CloudFormation Template: BaseNetwork

In our Aviator *service level* we assist customers with creating custom CloudFormation templates to describe their environments. For customers at both the Navigator and Aviator service levels we make a standardized CloudFormation Template, BaseNetwork, available to create the initial network and all of its necessary components. The rest of this section will describe the elements that are part of the BaseNetwork CloudFormation Template, and their associated components. The BaseNetwork template can be downloaded from `here`.

Parameters

- VPCCIDR - CIDR for the VPC
- SubnetPublicAZ1 - CIDR for Public subnet
- SubnetPublicAZ2 - CIDR for Public subnet
- SubnetPrivateAZ1 - CIDR for Private subnet
- SubnetPrivateAZ2 - CIDR for Private subnet
- InstanceTenancy - Single or Multi-Tenant Hypervisor
- Environment - Dev, Test, Prod etc.

Networking

- The CloudFormation template has two major options:
  - 2 Availability Zones with 4 Subnets
  - 3 Availability Zones with 6 Subnets
- Defaults to using CIDR: 172.18.0.0/16
  - Public Ranges
    - 172.18.0.0/22 - 1,022 Hosts - Public AZ1
    - 172.18.4.0/22 - 1,022 Hosts - Public AZ2

        - 172.18.8.0/22 - 1,022 Hosts - Public AZ3
        - 172.18.12.0/22 - 1,022 Hosts - Public AZx
        - 172.18.16.0/20 - 4,094 Hosts - Additional public (4 more public with size listed above)
    - Private Ranges
        - 172.18.32.0/21 - 2,046 Hosts - Private AZ1
        - 172.18.40.0/21 - 2,046 Hosts - Private AZ2
        - 172.18.48.0/21 - 2,046 Hosts - Private AZ3
        - 172.18.56.0/21 - 2,046 Hosts - Private AZx
        - 172.18.64.0/18 - 16,382 Hosts - - Additional private (8 more private using size above)
    - 172.18.128.0/17 - 32,766 Hosts - Special needs (16 more private using size above)
- Route Tables
    - RouteTablePublic - route table for Public subnets
    - RouteTablePrivateAZ1 - route table for Subnet Private AZ1
    - RouteTablePrivateAZ2 - route table for Subnet Private AZ2
- Default Gateways
    - Internet Gateway (IGW) - Default GW for the Public Subnets
    - ASGNatAZ1 Instance ID - Default GW for Subnet Private AZ1
    - ASGNatAZ2 Instance ID - Default GW for Subnet Private AZ2
    - ASGNatAZ3 Instance ID - Default GW for Subnet Private AZ3 (if necessary)

HA NAT

- High Availability NAT gateways get created in the public subnets (1 per AZ)
    - NatAZ1
    - NatAZ2
    - NatAZ3 (if necessary)

Tags

- Service Provider - “Rackspace”
- Environment - from Parameter Environment
- Name - Resource name (e.g. IGWBase, SubnetPublicAZ2)

Outputs

- outputVPCID
- outputSubnetPublicAZ1
- outputSubnetPublicAZ2
- outputSubnetPublicAZ3 (if necessary)
- outputSubnetPrivateAZ1
- outputSubnetPrivateAZ2
- outputSubnetPrivateAZ3 (if necessary)

6.2 Virtual Private Cloud (VPC)

Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the Amazon Web Services (AWS) Cloud where you can launch AWS resources in a virtual network that you define. Click here to learn more about VPC.

6.2.1 Rackspace Base Network VPC

For most Fanatical Support for AWS customers, Rackspace recommends the deployment of a single VPC per AWS account to provide operational simplicity while meeting stringent security requirements. Segregation will be accomplished by creating Public and Private subnets, and by relying on carefully created Security Groups that only allow the required granular access (further details in the *Security section*).

If further segregation were required to control access (e.g. Production vs. Test vs. Development), Rackspace's recommendation is to create a separate AWS accounts, and **not** a separate VPC in the same AWS account. This is because a second VPC in the same AWS account does not provide control plane isolation of resources, and could complicate ongoing operational processes.

You can assign a single CIDR block to a VPC. The allowed block size is between a /28 netmask and /16 netmask. In other words, the VPC can contain from 16 to 65,536 IP addresses. You cannot change the size of a VPC after you have created it. If your VPC is too small to meet your needs, you will need to create a new, larger VPC, and then migrate your instances to the new VPC.

The VPC requires an RFC 1918 CIDR range (Private addresses). The default is a /16 network, however, you need to carefully consider the range you select to ensure the range does not overlap with your other environments (on premise, other AWS VPCs, Rackspace dedicated environments, etc.).

The CloudFormation template captures the CIDR range in the Parameter: VPCCIDR. Detailed IP addressing recommendations will be discussed in the *Subnets section*. The BaseNetwork CloudFormation template's default VPC CIDR is 172.18.0.0/16.

6.3 Availability Zones (AZs)

Each region contains multiple distinct locations called Availability Zones, or AZs. Each Availability Zone is engineered to be isolated from failures in other Availability Zones, and to provide inexpensive, low-latency network connectivity to other AZs in the same region.

An Availability Zone (AZ) is one or more data centers in close geographic proximity connected together over low-latency/high-speed links.

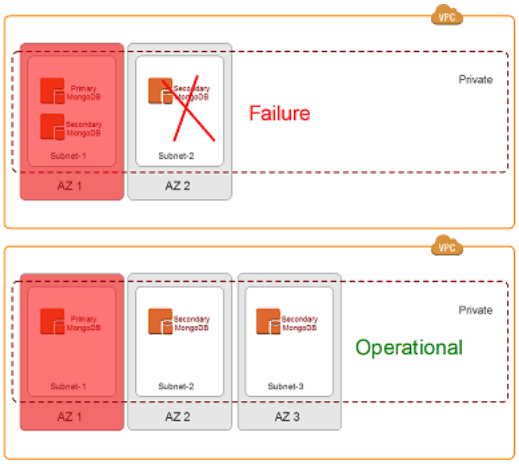
By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location. Note: Each AWS region provides a minimum of two AZs.

6.3.1 Rackspace Availability Zone Recommendations

Rackspace typically recommends a two AZ deployment, which provides availability and redundancy while reducing complexity, operational overhead, and cost.

There are situations where a third AZ may be required to address specific application-centric requirements:

- Example 1: MongoDB's Election and Quorum constraints require three AZs to survive a single AZ failure that contains the primary and a secondary in a three-node cluster.



Example: If AZ 1 were to fail with only two AZs, the MongoDB cluster would fail since Election and Quorum constraints were not achieved. With three AZs, Election and Quorum constraints are achieved and the MongoDB cluster remaines operational.

- Example 2: Applications that have strict load and availability requirements that cannot be met by relying on Auto Scaling Groups require over-provisioning. Adding a third AZ could be considered to reduce costs by lowering needed the over-provisioning.

Example: Strict application load and availability requirements dictates 12 servers to by up at all times, even if one AZ fails, (assuming AutoScale cannot scale fast enough during an AZ failure). This requires over provisioning. Adding more AZs to the architecture would reduce cost, but could potentially add complexity.

6.4 Subnets

You can create a VPC that spans multiple Availability Zones. After creating a VPC, you can add one or more subnets in each Availability Zone. Each subnet must reside exclusively within one Availability Zone and cannot span zones. AWS assigns a unique ID to each subnet.

If a subnet's traffic is routed to an Internet gateway, the subnet is known as a Public subnet. If the instance in a Public subnet needs to communicate with the Internet, it must have a public IP address or an Elastic IP address. If a subnet doesn't have a direct route to the Internet gateway, the subnet is known as a Private subnet.

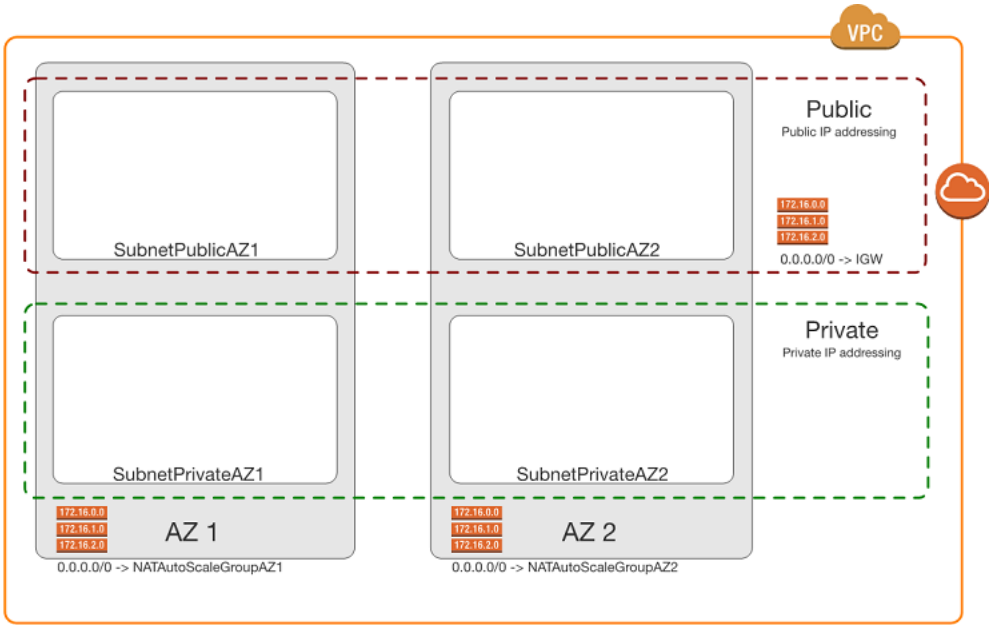
When you create a subnet, you specify the CIDR block for the subnet. The CIDR block of a subnet can be the same as the CIDR block for the VPC (for a single subnet in the VPC), or a subset (to enable multiple subnets). The allowed block size is between a /28 netmask and /16 netmask. You can create more than one subnet in a VPC, but the CIDR blocks of the subnets must not overlap.

6.4.1 Rackspace Subnet Recommendations

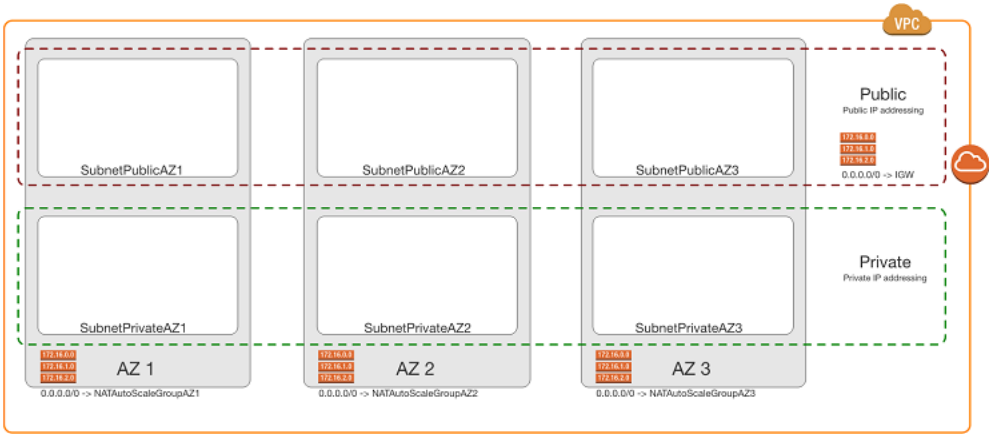
For most deployments, Rackspace recommends having two tiers of Subnets: Public and Private.

- EC2 instances in Public Subnets have public IP addresses associated with them and have a direct route to an AWS Internet Gateway (IGW), thus having the capability (if required) to access or be accessed by the Internet.
- EC2 instances in Private Subnets only have private IP addresses and cannot be accessed by the Internet. These EC2 instances have the capability to access the Internet via a NAT Gateway in the Public subnets (further info in the *NAT section*).

Assuming a typical two AZ deployment, four subnets would be required (two for Public and two for Private).



In situations where a third AZ is required (e.g. MongoDB servers in the Private subnets) then six subnets would be required (three for Public and three for Private).



It is important to note that within each tier, all the subnets will have the same network mask to simplify the operational processes (e.g. /22 for all Public subnets and /21 for all Private subnets).

Unlike traditional networking segmentation approaches that requires separate subnets (VLANs) for web, batch, application, and data tiers, AWS's use of Security Groups allows you to leverage just the Public and Private subnets, applying specific Security Groups to each tier (further info in the *Security section*). Thus a deployment would looks like:

- Public Subnets
  - Bastion servers

  - NAT servers (if not using a NAT Gateway)
  - VPN servers (if not using a Virtual Private Gateway)
  - Web servers not behind any ELB
- Private Subnets
  - Web servers behind an ELB
  - Batch-tier instances
  - App-tier instance
  - Data-tier instances

The CloudFormation template uses the built-in "GetAZs" function to map the first or second AZ to the specified subnet in a particular region (e.g. us-west-1a and us-west-1b). The CloudFormation template also captures the CIDR range for the subnet in the parameters:

- SubnetPublicAZ1 - CIDR for Public subnet
- SubnetPublicAZ2 - CIDR for Public subnet
- SubnetPrivateAZ1 - CIDR for Private subnet
- SubnetPrivateAZ2 - CIDR for Private subnet
- SubnetProtectedAZ1 - CIDR for Protected subnet
- SubnetProtectedAZ2 - CIDR for Protected subnet

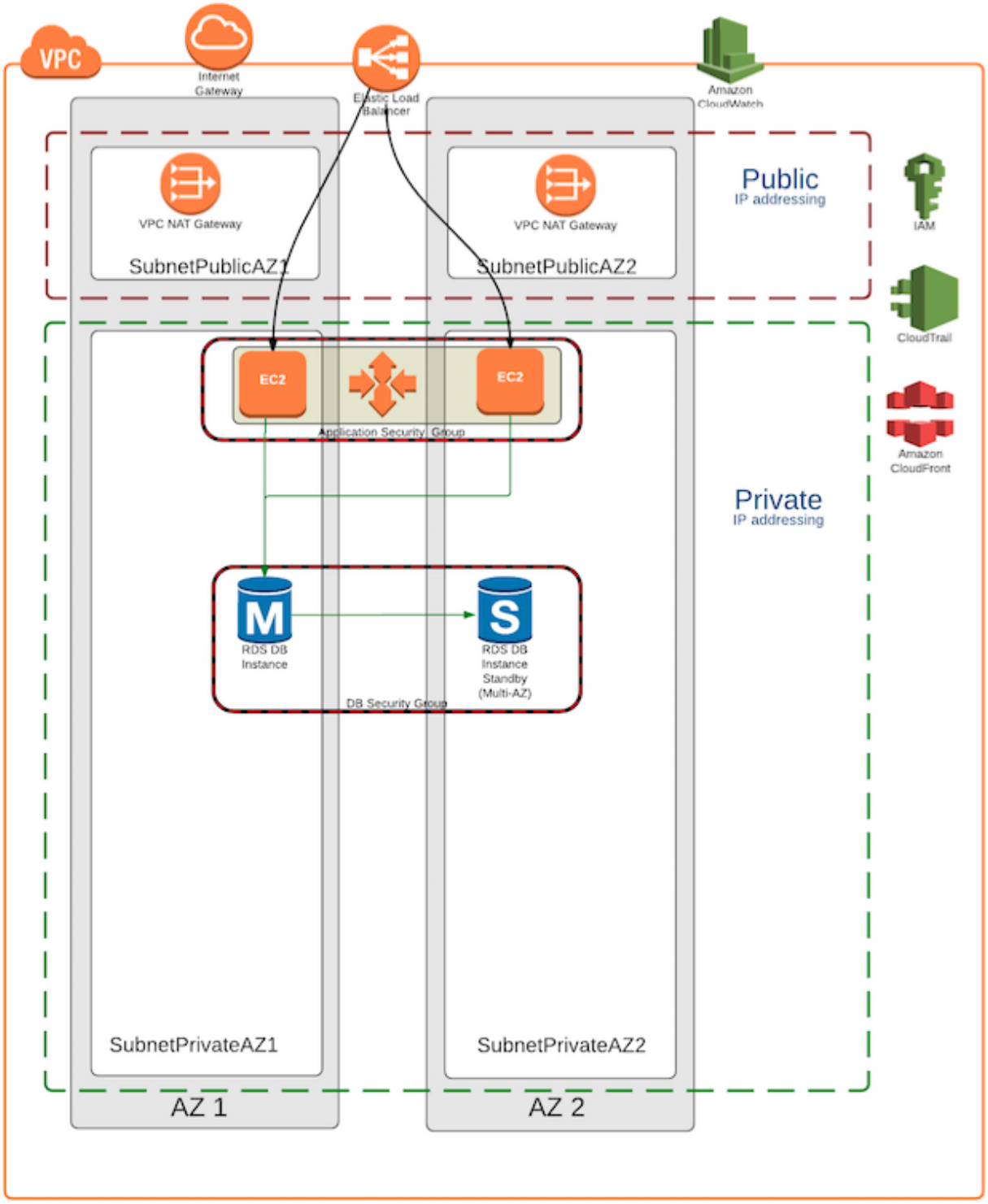
It is recommend that you choose the CIDRs carefully to map with the applications' requirements; however, most AWS customers typically allocate roughly double the IP addresses for private subnets than public subnets. The default CIDRs in the BaseNetwork CloudFormation template are detailed in the *CloudFormation section*.

6.5 Highly Available Network Address Translation (HA NAT)

In a VPC, you can use private subnets for instances that do not require directly accessible Internet-facing IP addresses. Instances in a private subnet can access the Internet without exposing their private IP address by routing their traffic through a Network Address Translation (NAT) gateway in a public subnet. Each NAT gateway is created in a specific Availability Zone (AZ) and has built-in redundancy in that AZ.

6.5.1 Rackspace NAT Recommendations

As mentioned above, NAT gateways are required for instances in a Private subnet to access the Internet. In the recommended *two AZ deployment*, Rackspace recommends leveraging one NAT gateway in each AZ - not sharing a NAT gateway with more than one AZ.



NAT gateways are created via the CloudFormation template which:

- Creates an Elastic IP Address (EIP) for each NAT gateway to be reachable on public networks
- Creates a route for each private network in each AZ to route all traffic through the corresponding NAT gateway in each AZ.

Rackspace does not recommend creating resources in one AZ that rely exclusively on a NAT gateway in a different AZ. In the event of a NAT gateway failure, resources in any AZ that depend on that single NAT gateway will be unable to access the Internet.

6.5.2 Migrating from a NAT instance

If you were previously using NAT instances for allowing resources on private networks to access the Internet, you should create a NAT gateway in each AZ, and change the routing tables for your private networks to use the new NAT gateway. Then, existing resources associated with your NAT instances (autoscale groups, NAT instances in EC2) can be removed; the change will only impact connections open at the time of the change to the routing table.

You should also take care to ensure that any existing whitelist of IPs from the NAT instances are also adjusted to reflect the new IPs of your NAT gateways.

6.6 Security

AWS provides a scalable, highly reliable platform that helps customers deploy applications and data quickly and securely.

When customers build systems on the AWS infrastructure, security responsibilities are shared between the customer and AWS. This shared model can reduce the customer's operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate. In turn, the customer assumes responsibility and management of the guest operating system (including updates and security patches), other associated applications, as well as the configuration of the AWS-provided security group firewall.

The Rackspace Fanatical Support for AWS offering takes some of the security burden from the customer by leveraging AWS security best practices and providing additional security capabilities. These include using/enabling Security Groups, Config, CloudTrail, CloudWatch, etc. In this section, we will focus on Security Groups.

6.6.1 Security Groups

A Security Groups acts as a virtual firewall that controls inbound and outbound traffic for one or more instances. When an instance is launched in a VPC, the instance can be assigned up to five security groups that are associated to the VPC. Specific inbound and outbound rules are then added to each security group that allows defined traffic to or from its associated instances. Rules can be modified at any time; the new rules are automatically applied to all instances that are associated with the security group. Note: by default, outbound rules allow all traffic to egress the instance and inbound rules allow nothing (implicit deny).

Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in a VPC could be assigned to a different set of security groups, thus easily creating isolation within the same subnet. When AWS is deciding whether to allow traffic to reach an instance, all the rules from all the security groups that are associated with the instance are evaluated at the same time. This is very different to the way Network ACLs (NACLs) work.

6.6.2 Network ACLs (NACLs)

AWS also offers network ACLs with rules similar to your security groups. NACLs act as a firewall for associated subnets, controlling both inbound and outbound traffic at the subnet level. The following summarizes the basic differences between security groups and network ACLs:

Security Group

- Operates at the instance level (first layer of defense)

- Supports allow rules only
- Is stateful: Return traffic is automatically allowed, regardless of any rules
- We evaluate all rules before deciding whether to allow traffic
- Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on

Network ACL

- Operates at the subnet level
- Supports allow rules and deny rules
- Is stateless: Return traffic must be explicitly allowed by rules
- We process rules in number order when deciding whether to allow traffic
- Automatically applies to all instances in the subnets it's associated with (backup layer of defense, so you don't have to rely on someone specifying the security group)

6.6.3 Rackspace Security Model

As a general best practice, Rackspace advises customers to use Security Groups as their primary method of securing workloads within AWS. While Network ACLs (NACLs) are typically more familiar to networking engineers, they often introduce complexity into AWS architectures.

Security Groups provide more granular control, are stateful (therefore more intelligent in allowing appropriate traffic) and apply only to the instance level. By using NACLs as well as Security Groups, you must consider all traffic in a stateless context (specifying inbound and outbound ports, including any ephemeral ports used by a given application) and these rules are applied at a subnet level; the "blast radius" or potential for impact when a NACL is incorrect or changed is significantly higher, without providing any tangible benefit over the use of a Security Group.

Rackspace and AWS recommend avoiding NACLs due to potential conflicts with Security Groups and performance degradation. If there are compliance requirements (e.g. PCI) that specifically call for NACLs, they will be used sparingly and with coarse controls to mitigate potential issues.

6.7 Tagging

AWS customers use tags to organize their EC2 resources (instances, images, load balancers, security groups, and so forth), RDS resources (DB instances, option groups, and more), VPC resources (gateways, option sets, network ACLS, subnets, and the like), Route 53 health checks, and S3 buckets. Tags are used to label, collect, and organize resources and become increasingly important as customers use AWS in larger and more sophisticated ways.

For example, customers can tag relevant resources and then take advantage *cost allocation via tagging*.

6.7.1 Rackspace CloudFormation Tagging

The BaseNetwork CloudFormation template makes use of tagging to drive many of the operational functions associated with the Fanatical Support for AWS offering. These include:

- Service Provider - "Rackspace"
- Environment - from Parameter Environment
- Name - Resource name (e.g. IGWBase, SubnetPublicAZ2)

CHAPTER

SEVEN

BILLING

When you signup for Fanatical Support for AWS, Rackspace establishes one or more AWS accounts for you and becomes your reseller of AWS services. This means that all billing of both infrastructure and support charges is provided through a consolidated Rackspace bill, and you do not have to maintain a payment relationship with AWS directly. The credit card you provided when signing up for your Rackspace account will be automatically billed for both your AWS infrastructure and support charges, as described below.

7.1 Billing Cycles

AWS bills for all infrastructure on a calendar month basis. AWS charges for the previous month's usage are typically finalized by the 10th day of each month. After the charges are finalized by AWS, both infrastructure and support charges are added to your Rackspace account and will appear on your next Rackspace bill. Each line item will include the month in which the charges were incurred. Your Rackspace bill is created the 15th of each month if you signed up for a Fanatical Support for AWS Account after July 6, 2017. If you signed up prior to July 6, 2017 or are using an account originally created for the Rackspace Public Cloud, you will be billed based on the anniversary date the account was created.

7.2 Financial Benefits of your Rackspace account

Your Rackspace account is the top-level container which contains one or more AWS accounts. When aggregating the usage to generate a bill at the Rackspace account level, you receive the following benefits:

- Favorable *Reserved Instance allocation*
- Tiering of usage across all accounts for AWS services which provide tiered pricing (for example, if S3 has a 0-10 TB storage tier and a 10-20 TB storage tier and you have one AWS account which uses 8 TB and another which uses 3 TB your overall usage would be rated as a combined 11 TB)

7.3 Monthly Service Fees

Your monthly service fees will be calculated by pooling the AWS infrastructure charges from all of your AWS accounts at the same service level. If the AWS infrastructure charges are greater than $30, a monthly service fee will be assessed and proportionally distributed out to each of your AWS accounts at that service level.

For example, if you have two AWS accounts at the Aviator service level, one with $45,000 in AWS infrastructure charges and the other with $30,000 in AWS infrastructure charges, your monthly service fee would be $25,000 based on your combined AWS infrastructure charge of $75,000.

During your first month with any AWS accounts at a specific service level, we will prorate the monthly service fee for the remainder of the month based on your signup date unless you make any Reserved Instance purchases during that month. If you do make a Reserved Instance purchase, we remove the proration as the service fees paid during your first month will cover our services for the length of the reserved instance reservation. Please note that proration is based on AWS account signup date, and not the date AWS infrastructure charges reach more than $30. For example, if your AWS account has a signup date of March 15th but you do not start using resources in your account until April 5th, you will be charged a full month of service fees for April.

7.4 Usage

The Usage page in the Fanatical Support for AWS Control Panel will provide you with a mid-month view of your charges and an estimate of your full month's charges, typically updated a few times per day, along with historical usage from previous months. You can use this information to avoid unexpected AWS infrastructure charges. To access the report, select the Usage link in the primary navigation.

Due to our *Account Defaults* and associated management tooling, each AWS account that you provision will have approximately $5-10 of monthly infrastructure charges, regardless of whether you provision any additional AWS resources.

7.5 Viewing your Invoices

To view your invoices, login to the Fanatical Support for AWS Control Panel and click the Billing link toward the top right of the page.

The primary account holder will receive an email any time a payment is processed, indicating that a new invoice is available for review.

7.6 Tagging

Rackspace will provide detailed views of your AWS billing data by resource tags. Tags must use a key from the following list (case sensitive) in order to be included in these views:

- BusinessUnit
- Group
- Department
- CostCenter
- Application
- Environment
- Project
- Owner
- Service
- Cluster
- Role
- Customer

- Version
- Billing1
- Billing2
- Billing3
- Billing4
- Billing5

We also include the following AWS-generated tags in the detailed views of your AWS billing data:

- aws:autoscaling:groupName
- aws:cloudformation:logical-id
- aws:cloudformation:stack-id
- aws:cloudformation:stack-name

While you may use tags outside of those listed above to identify your resources for other reasons, they will not be included in the detailed views of your billing data.

7.7 Modifying your Payment Method

If you need to update the credit card or ACH (eCheck - United States only) details that you have on file, login to the Fanatical Support for AWS Control Panel and click the Billing link toward the top right of the page. From there, you'll find a link to update your payment details.

CHAPTER

EIGHT

RESERVED INSTANCES

Reserved Instances play an important role in helping you manage the overall costs of your AWS environments. In cases where you plan to have sustained 24x7 usage of one or more EC2 or RDS instances of the same instance type in the same availability zone and region, purchasing a one-year or three-year reserved instance can save up to 70% versus the on-demand hourly rates.

You can learn more about Reserved Instances at https://aws.amazon.com/ec2/purchasing-options/reserved-instances/.

8.1 Allocation across AWS accounts

If you have more than one AWS account that is part of the same Rackspace account you can benefit from automatic allocation of unused reserved instances from one AWS account to another. Our billing system automatically detects unused reserved instances on your AWS account and searches for corresponding EC2/RDS on-demand instances of the same instance type and provisioned in the same availability zone on your other AWS accounts under the same Rackspace account. If a match is found, the reserved instance is automatically applied to the usage on the other AWS account.

A few key considerations:

- Reserved instances are a billing construct only; you do not need to specify anything at the time you launch the EC2 or RDS instance. As long as the instance type and availability zone match, the reserved instance benefit will automatically be applied.
- The allocation of reserved instances to other accounts occurs on an hourly basis; therefore, if the account that purchased the reserved instance originally begins using the same instance type in the same availability zone at a later time, the reserved instance benefit will be applied to the original purchasing AWS account.
- Although the underlying data centers powering a specific availability zone vary across accounts (e.g. us-east-1a may be different on account X than account Y), for the purposes of reserved instance allocation the availability zone match is done based only on the availability zone name.

8.2 Purchasing Reserved Instances

You can purchase reserved instances directly from AWS via the AWS Console or CLI, or programmatically via the SDKs or CLI.

8.3 Impact on Monthly Service Fees

As described in the *Pricing* section, your monthly service fee is calculated based on the total of all AWS infrastructure charges. Reserved instance purchases are included in these charges, so months where you make one or more reserved

instance purchases may cause you to incur a higher monthly service fee. Since the reserved instance will lower or eliminate your charges for that portion of the infrastructure in future months, you'll see a similar benefit apply to your monthly service fees. The effective rate of our service fees decreases as AWS infrastructure spend increases, so you will likely pay a lower total amount of service fees over the life of the reserved instance than if you had run on-demand instances during the same time period.

8.3.1 Example 1: Navigator Service Level

For this example, assume that your application requires a single m4.4xlarge instance and no other AWS on-demand infrastructure. Your options are:

- Reserved Instance Purchase
  - Reserved Instance Purchases - month 1 (one m4.4xlarge instance; one year; all up front): $5,082
  - Service fee - month 1 (based on spend of $5,082): $750
  - Service fee - months 2 through 12 (based on spend of $0): $0
  - Total cost: $5,832
- No Reserved Instance Purchase
  - On-demand usage - months 1 through 12 (one m4.4xlarge instance): $735.84
  - Service fee - months 1 through 12 (based on spend of $735.84): $400
  - Total cost: $13,630.08

In the example above, you would save 84% on service fees and 42% on AWS infrastructure costs when purchasing reserved instances.

8.3.2 Example 2: Aviator Service Level

For this example, assume that your application requires 25 c4.2xlarge instances and no other AWS on-demand infrastructure. Your options are:

- Reserved Instance Purchase
  - Reserved Instance Purchases - month 1 (25 c4.2xlarge instances; one year; partial front): $33,648.25
  - Service fee - month 1 (based on spend of $33,648.25): $17,000
  - Reserved Instance monthly fees - months 2 through 12 (25 c4.2xlarge instances): $2,573.25
  - Service fee - months 2 through 12 (based on spend of $2,573.25): $2,500
  - Total cost: $106,454
- No Reserved Instance Purchase
  - On-demand usage - months 1 through 12 (25 c4.2xlarge instance): $8,048.25
  - Service fee - months 1 through 12 (based on spend of $8,048.25): $5,750
  - Total cost: $165,579

In the example above, you would save 36% on service fees and 36% on AWS infrastructure costs when purchasing reserved instances.

8.3.3 Example 3: Navigator Service Level

For this example, assume that your application requires 15 c4.2xlarge instances and no other AWS on-demand infrastructure. Your options are:

- Reserved Instance Purchase
  - Reserved Instance Purchases - month 1 (15 c4.2xlarge instances; one year; partial front): $20,188.95
  - Service fee - month 1 (based on spend of $20,188.95): $3,500
  - Reserved Instance monthly fees - months 2 through 12 (15 c4.2xlarge instances): $1,543.95
  - Service fee - months 2 through 12 (based on spend of $1,543.95): $750
  - Total cost: $48,922.40
- No Reserved Instance Purchase
  - On-demand usage - months 1 through 12 (15 c4.2xlarge instance): $4,828.95
  - Service fee - months 1 through 12 (based on spend of $4,828.95): $1,500
  - Total cost: $75,947.40

In the example above, you would save 35% on service fees and 36% on AWS infrastructure costs when purchasing reserved instances.

Note: The costs used in the examples above are for illustrative purposes only and could change at any time. In some cases, your service fees will be higher when purchasing reserved instances versus not purchasing them, decreasing, but not eliminating, your overall savings. Your *Account Manager* can assist you with calculating reserved instance benefits for your specific account.

8.4 Additional Considerations

8.4.1 Reserved Instance utilization is calculated hourly

As noted above, Reserved Instances are a billing construct within AWS. A Reserved Instance purchase provides you a discounted hourly rate for the instance type and operating system specified at purchase. This discount is granted per billing hour, which could inadvertently lead to additional On-Demand infrastructure charges if you are not careful. Any time an instance is launched, or transitions from Stopped to Running state, a new billing hour begins and a full hour of usage is immediately charged. This occurs even if the same instance is Stopped and returned to the Running state multiple times in the same hour.

Note: The above behavior applies to Reserved Instances as well as On-Demand instances. For additional details, see: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-lifecycle.html

8.4.2 The Billing Hour is unique for each instance

The billing hour for each instance is tied to when that instance last entered the Running state, so it is likely that your billing hour will be different for each of your instances. If you launch an instance at 9:05 AM, the billing hour for that instance starts at 5 minutes past each hour, unless it is Stopped or Terminated. If the instance is Stopped, the next time it enters the Running state a new billing hour begins (with a new hourly start time) and you will immediately be charged for 1 hour of usage. This behavior occurs regardless of your usage of Reserved Instances, so it's important to understand how this may affect your bill.

- Example 1 - Rebooting an instance multiple times per hour: Assume that you have purchased a single Reserved Instance. You reboot your instance via the AWS console or CLI multiple times. Because reboots do not trigger a new billing hour, you are not charged for additional usage.
- Example 2 - Stopping or Terminating an instance and relaunching within the billing hour: Assume that you have purchased and launched a single Reserved Instance. 10 minutes into the billing hour, you stop or terminate the instance, and then relaunch the instance within the same billing hour. This activity causes a new billing hour to start, and thus, you are immediately charged for 1 hour of usage. You are not refunded or credited for any remaining minutes from the old billing hour. Since you have only purchased a single Reserved Instance, and have had two billing hours overlap, you will have consumed your 1 Reserved Instance hour for that time period, and will also be charged 1 hour at the On-Demand rate. If the instance remains running, it will begin using the reserved instance rate during the next billing hour.
- Example 3 - Stopping or Terminating an instance and relaunching during the next billing hour: Assume that you have purchased and launched a single Reserved Instance. 10 minutes into the billing hour, you stop or terminate the instance. You wait 51 minutes and then relaunch the instance. When the instance enters the Running state, a new billing hour starts. Since you have waited until the previous billing hour completed, you now have a Reserved Instance reservation available. You are not charged for any additional On-Demand usage.
- Example 4: Oversubscribing your Reserved Instances for half the time: Assume that you have purchased 2 Reserved Instances. You decide to launch 4 instances for only 12 hours per day, and Terminate or Stop them for the other 12 hours. In this scenario, you will be charged an additional 24 hours (12 hours x 2 instances) at the On-Demand rate, as you already have 2 instances consuming your Reserved Instance reservations and any additional instances running at the same time are charged at the On-Demand rate. You cannot store unused Reserved Instance hours and use them to oversubscribe at other times. If you have workloads that need predictable scaled capacity, you may consider want to consider a purchase of Scheduled Reserved instances instead: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-scheduled-instances.html

CHAPTER

NINE

ACCESS AND PERMISSIONS

Controlling access and permissions to the Rackspace and AWS control planes (APIs and UIs) along with the resources you deploy at AWS are a critical part of the overall security of your environment. This section outlines several core concepts related to access and permissions, along with details on how to grant members of your team and others access to your account, as needed.

9.1 User Management and Permissions

9.1.1 Account Owner

When you sign up for Fanatical Support for AWS, the first user you create is the **Account Owner**. After signing up, you can reassign Account Owner status to another user on the account. You can make this change from the **Account Settings** page. There can only be one Account Owner at a time.

The **Account Owner** has full administrative privileges, including:

- AWS `AdministratorAccess` IAM policy rights on all AWS accounts
- `Admin` rights to all Fanatical Support for AWS features on all AWS accounts
- `Admin` rights to the Rackspace Billing and Payments portal
- Ability to add additional AWS accounts for Rackspace to manage
- Ability to create and delete users on the Rackspace account and manage their permissions on a per AWS account basis
- Ability to make other users Account Administrators
- Ability to reassign Account Owner status
- Ability to configure Rackspace account-wide settings including enabling multi-factor authentication, configuring session duration, etc.
- Ability to cancel the Rackspace account

9.1.2 Creating and Managing Users

If you have more than one person in your organization that will need access to the Fanatical Support for AWS Control Panel, the AWS Console/APIs, or both, you can create additional users and assign them permissions on a per AWS account basis.

To create and manage users:

1. Log in to the Fanatical Support for AWS Control Panel

2. Click your **user name** at the top right to activate the account menu
3. Select **User Management**

From the **User Management** page you'll have the ability to create new users, manage existing users, and assign permissions to users.

9.1.3 Account Administrator

One permission that can be assigned is **Account Administrator**. This permission is useful when the Account Owner wishes to delegate a significant set of rights to another user on the account, e.g. if someone else in the organization besides the Account Owner is responsible for creating new users and assigning them permissions.

Users with the **Account Administrator** right have the following privileges:

- AWS `AdministratorAccess` IAM policy rights on all AWS accounts
- `Admin` rights to all Fanatical Support for AWS features on all AWS accounts
- `Admin` rights to the Rackspace Billing and Payments portal
- Ability to add additional AWS accounts for Rackspace to manage
- Ability to create and delete users on the Rackspace account and manage their permissions on a per AWS account basis

Account Administrators do **NOT** have the following permissions:

- Ability to view or modify the Account Owner or other users with the Account Administrator permission
- Ability to make other users Account Administrators
- Ability to configure Rackspace account-wide settings including enabling multi-factor authentication, configuring session duration, etc.
- Ability to cancel the Rackspace account

9.1.4 Understanding and Managing Permissions

The **Account Owner** and **Account Administrators** have the ability to manage permissions for other users.

There are two categories of permissions:

1. Rackspace Account Permissions

These permissions are Rackspace account-wide and broader than Fanatical Support for AWS.

- **Account Administrator** - Gives the user a substantial subset of Account Owner permissions (see above for details).
- **Billing and Payments** - Provides access to the Rackspace Billing and Payments portal which includes information like invoices, payment methods, and billing settings.
- **Support Tickets** - Provides the ability to give more granular access to your Rackspace Account support tickets. You can prevent users from seeing tickets. You can also allow users to only see tickets, however, they will not be able to create tickets.

2. Product Permissions

These permissions are Rackspace product specific. This is where Fanatical Support for AWS permissions are managed (other product permissions will not be covered in this guide).

There are three Fanatical Support for AWS permissions:

- **Allow this user to add AWS Accounts** - Enables the user to add additional AWS accounts for Rackspace to manage. These could be new or existing AWS accounts.
- **Fanatical Support for AWS** - Controls what access, if any, the user will have within the Fanatical Support for AWS Control Panel. This permission applies to **all** Rackspace features including Passport, Logbook, Compass, and Usage. This permission is configured on a per AWS account basis.
- **AWS Console and APIs** - Controls what access, if any, the user will have when federating to the AWS Console or retrieving AWS temporary API credentials. This permission can be any AWS managed or custom IAM policy available on the AWS account and is configured on a per AWS account basis.

Note: It is possible that federation from the Fanatical Support for AWS Control Panel to AWS may fail for IAM policies that are long. Specifically, we leverage the Security Token Service `AssumeRole` call to dynamically scope access and AWS limits the Policy parameter to a max of 2048 bytes. The AWS `ReadOnlyAccess` managed IAM policy is one such policy that exceeds the length, so Rackspace provides a `RackspaceReadOnly` policy that works. If you have trouble federating to AWS due to IAM policy length, please contact your Fanatical Support team for assistance.

Rackspace Permission Types

Rackspace specific permissions can be set to one of three values:

- **None** - No access
- **Observer** - Read-only access
- **Admin** - Read and write access

9.1.5 Permission Example

You have 2 AWS accounts managed by Rackspace, both at the Aviator service level. They are named **App1-Staging** and **App1-Production**.

You might grant a junior developer working on this application the following permissions:

Account Permissions

- **Account Administrator** - `Disabled`
- **Billing and Payments** - `None` since he does not need access to invoice and payment information

Product Permissions

- **Allow this user to add AWS Accounts** - `Disabled`

App1-Staging

- `Admin` access to **Fanatical Support for AWS** so, for example, he has the ability to authenticate to instances via Passport.

- `AdministratorAccess` IAM policy access so he has full access to AWS services via the **AWS Console and APIs**.

App1-Production

- `Observer` access to **Fanatical Support for AWS** so he can view but not make changes to the production AWS Account via Rackspace tooling. This will disable Passport access but Compass and Logbook are still available.
- `RackspaceReadOnly` IAM policy to limit his **AWS Console and API** access to read-only.

9.2 Rackspace Account

Your Rackspace account is the top-level container which contains one or more AWS accounts. All user and permissions management takes place at the Rackspace account level, though you can limit specific users on your account to only have access to specific AWS accounts. The Rackspace account is also used for billing purposes. All charges from each of the AWS accounts are *aggregated at the Rackspace account level*.

9.3 AWS Console

Once you have logged in to the Fanatical Support for AWS Control Panel you will see a listing of all AWS accounts you have access to.

If you wish to access the AWS Console you can click the "Log in to AWS Console" button and you will be automatically signed in as a federated user. This allows you to maintain one set of credentials to access both the Fanatical Support for AWS Control Panel and the AWS Console. As described in the *User Management and Permissions section* the access a user will receive when they federate to the AWS Console will be determined by the AWS IAM Role selected when configuring the user's permissions.

9.4 AWS CLI, SDKs, and APIs

There are two methods for accessing the AWS command-line interface (CLI), software development kits (SDKs), and application programming interfaces (APIs):

1. From the Fanatical Support for AWS Control Panel, navigate to the Account Details screen for the AWS account you would like to access and click the View Credentials button. You will be issued AWS Security Token Service (STS) credentials that are valid for up to 60 minutes and are scoped to the same level of permissions as if you were to federate to the AWS Console. This is the preferred method of short-lived, infrequent access as access to the credentials is tied to your Fanatical Support for AWS user and is logged in the *Rackspace Logbook*.
2. If you require longer-lived, more persistent access to the CLI, SDKs, or APIs you should create an IAM user with access keys (if the access will be from a user's workstation) or an IAM instance role (if the access will be from resources, such as EC2 instances, running at AWS). Note that directly-created IAM users or roles are not managed within the Fanatical Support for AWS user management system, and therefore modifying or terminating access must be done directly within AWS IAM.

If you need assistance determining which option is best for your specific use case, please *contact a Racker*.

9.5 AWS Identity and Access Management (IAM)

As described earlier, our standard best practice is to manage all access as either:

- Users within the Fanatical Support for AWS Control Panel
- IAM Roles for AWS resources, such as EC2 instances, requiring access to other AWS services

Occasionally, a use case will arise where it is necessary to directly create an IAM user or role. These scenarios typically involve a third-party tool or SaaS needing access to your account, such as a continuous integration and deployment system like CircleCI or a local file management application that integrates with S3 such as Cyberduck. If you must create a user or role directly within IAM, please remember the following:

- The IAM policy that you assign should be created to allow the minimum level of access required to your AWS account. If you need assistance with creating the appropriate IAM policy, please *contact us*.
- IAM users and roles are managed outside of the Fanatical Support for AWS Control Panel and will not show up in the User Management system. Therefore, any modifications or revocation of access must also be performed directly within AWS IAM.
- A default IAM password policy is included in our *AWS account defaults*. We do not recommend weakening or disabling these requirements, as they are put in place to protect your account from brute-force password attacks.
- An IAM user should typically have password access or access keys, but not both. Password access is used for accessing the AWS Console (and most of these use cases should be covered under the Fanatical Support for AWS Control Panel permissions model) and access keys are used for programmatic access. In almost all cases where you are creating an IAM user, only access keys should be required.

For assistance in determining the appropriate method of granting access to your account, please *contact us*.

CHAPTER

TEN

SECURITY

10.1 Rackspace Shared Management Services

Rackspace takes the security of our shared management services and the Fanatical Support for AWS Control Panel extremely seriously. All infrastructure is deployed on AWS leveraging the same set of best practices that we apply to customer accounts. The following sections provide a sample of some of the key security focus areas.

10.1.1 Racker Authentication

All Rackspace employees must leverage two-factor authentication for all access to customer account data and customer environments.

10.1.2 Racker Privileges

The level of privileges each Racker has to our Fanatical Support for AWS management systems is tightly controlled based on job role and is periodically reviewed to ensure that each Racker has the minimum level of permissions required to adequately perform their job duties. All privilege changes require management approval and are also logged for later review.

10.1.3 Encryption at Rest

All databases leverage the AWS Key Management Service (KMS) for data encryption at rest. All EBS volumes are encrypted with KMS in addition to application-level encryption of secrets using KMS and the AWS SDKs.

10.1.4 Encryption in Transit

All communication between services that make up the Fanatical Support for AWS shared management system are encrypted during transit using SSL. Our customer and Racker UIs and APIs are only accessible via HTTPS.

10.1.5 AWS Account Best Practices

As outlined in the *AWS Accounts section* we always enable AWS CloudTrail and AWS Config in all regions for each new account. We also have checks within *Rackspace Compass* that ensure these remain enabled and configured per our best practices.

10.1.6 Activity Logging

As described in the *Rackspace Logbook section* all control plane and data plane activities are logged and visible to both customers and Rackers via the Fanatical Support for AWS Control Panel, providing a complete playback of events that occurred on an account.

10.2 AWS Security

Amazon Web Services places a high degree of importance on the security of your infrastructure. For an overview of the AWS Security Processes we recommend reviewing their whitepaper.

We also encourage you to review the Securing Data at Rest with Encryption whitepaper for an overview of the methods for securing your data.

If you have questions regarding any aspect of the whitepapers or the security of your environment, please *contact a member of your Support team*.

10.2.1 Security Updates

As Amazon Web Services says on their Security Bulletins web page, "No matter how carefully engineered the services are, from time to time it may be necessary to notify customers of security and privacy events with AWS services."

If you are interested in staying informed about these security bulletins, the AWS Security Bulletins web page or the companion RSS feed are the appropriate channels to watch. From time to time, Rackspace will provide additional detail or guidance to our customers for specific security incidents. We will publish such value-add security updates below.

Rackspace Response to Meltdown and Spectre

On 3 January 2018, Rackspace was made aware of a vulnerability affecting certain processors by Intel, AMD and ARM. Multiple vendors have subsequently released statements regarding the vulnerability and its impact on their respective environments.

These issues were originally uncovered by Google's Project Zero. Their research findings show that an unauthorized party may read sensitive information in the system's memory such as passwords, encryption keys, or sensitive information open in applications.

The remainder of this update is addressed to our Fanatical Support for AWS customers specifically. For updates about our other Rackspace supported hosting environments, please refer to the Rackspace blog.

Overview

Details about the security vulnerabilities can be found in CVE-2017-5753, CVE-2017-5715, and CVE-2017-5754. Amazon's security bulletin regarding these vulnerabilities can be found here. The website spectreattack.com provides a good overview of these vulnerabilities.

There is not a single fix There is no single fix for all three security vulnerabilities. Many vendors have patches available for one or more of these attacks.

Amazon Web Services (AWS) Response These vulnerabilities affect many CPUs, including those used by Amazon EC2. As of 2018/01/04 15:30 PST, Amazon reports that "all instances across the Amazon EC2 fleet are protected...against these threats from other instances."

This is an important first step in mitigating the security risk that these vulnerabilities represent for your environments. As a result of this fix to the EC2 infrastructure, attackers will no longer be able to exploit EC2 infrastructure to view the contents of memory allocated to a different virtual machine running on the same hypervisor as their attacking software.

Actions that customers are responsible for taking Despite the actions taken by Amazon to patch EC2 infrastructure, a risk remains that malicious software running within your guest operating system may be able to exploit these security vulnerabilities to gain access to private data stored in memory on your EC2 instances. To protect against this risk, Rackspace will be communicating with you over the next several days with guidance and options for patching your EC2 instances.

Watch this page for updates

Please check back at this web page for updates on this security issue. Rackspace will post updates as additional details become available from affected vendors, and we will add additional guidance to this page over the next several days.

Change Log

Date	Description
2018/01/04 18:03 CST	Initial revision of this security update
2018/01/05 08:02 CST	Update title; Update status to reflect that Amazon have completed EC2 infrastructure patching
2018/01/04 14:57 CST	Minor update to wording to better align to Amazon's security bulletin

CHAPTER

ELEVEN

COMPLIANCE

11.1 PCI-DSS

Is Fanatical Support for AWS PCI-DSS compliant?

Rackspace is a certified Level 1 Payment Card Industry (PCI) Service Provider on Fanatical Support for AWS.

What was the scope of the PCI-DSS assessment?

For Fanatical Support for AWS, the Rackspace Service Provider assessment scope is detailed in the Executive Summary document provided to all customers. This assessment includes tooling and infrastructure operated by Rackspace and excludes AWS infrastructure, which is covered under their Report on Compliance.

Fanatical Support for AWS and related systems and tooling are PCI-DSS compliant. Does that mean that my solution will be compliant as well?

Hosting a solution with Rackspace does not make a customer PCI-DSS compliant. Fanatical Support for AWS Solution Architects are happy to assist our customers in navigating our product portfolio to identify solutions which meet their regulatory needs.

Can Rackspace help my solution become PCI-DSS compliant?

Rackspace is not a Qualified Security Assessor (QSA) and therefore cannot give a qualified opinion on the PCI-DSS compliance status of a customer's solution. In addition, due to many variations in our service delivery configurations we cannot offer PCI-DSS compliant solutions "out of the box." However, Rackspace can provide services, products and an extensive partner network that will satisfy many of the necessary PCI-DSS requirements. For a detailed list of controls and how Rackspace can assist, please request the PCI Responsibility Matrix.

Can Rackspace provide proof of its PCI-DSS compliance?

Rackspace can provide the following PCI-DSS Compliance Package:

- PCI Responsibility Matrix
- PCI-DSS Report on Compliance Executive Summary
- List of controls that belong to the Service Provider
- The Rackspace Attestation of Compliance

Note: Rackspace cannot release the full PCI-DSS Report on Compliance as it contains proprietary and commercially sensitive details of Rackspace security processes.

How can I get the PCI DSS Compliance Package?

Customers can access attestation of compliance forms in the Fanatical Support for AWS control panel: under the account drop-down in the upper right-hand corner, select "Documents and Forms", and navigate to the "Rackspace Cloud Security Documents" section.

Does Fanatical Support for AWS service level matter for PCI-DSS?

Both our Navigator and Aviator customers can leverage our PCI-compliant tooling and infrastructure. Note, however, that the Aviator service level provides a greater number of value-added services to include design, service selection, monitoring, and more that make achieving PCI-DSS compliance easier.

I have general questions about Rackspace Security beyond the scope of PCI-DSS - where can I get answers to those questions?

We have a Rackspace Information Security FAQ which includes additional information around security policy, internal organization, human resources, access controls, and more. Similar to the PCI-DSS Compliance Package, it can be requested from your Fanatical Support for AWS Technical Account Manager.

11.2 HIPAA

Can Fanatical Support for AWS support HIPAA workloads?

Yes, Rackspace can act as a business associate to support customers with HIPAA workloads at AWS.

Why is it important to have a Managed Service Provider (MSP) who can manage workloads on top of AWS if AWS already provides HIPAA-eligible services?

Any business that has needs to store, process, or transmit HIPAA data needs to ensure that the Managed Service Provider they choose on top of AWS has practices in place to allow them to comply with HIPAA, as well as a signed BAA (Business Associate Agreement).

Does Fanatical Support for AWS service level matter for HIPAA?

We provide management for both Navigator and Aviator customers running HIPAA workloads at AWS. However, with Aviator service level customers can take advantage of value-add services like best-practice architecture, service selection, patching, monitoring, and ongoing operations that may make achieving HIPAA compliance easier for customers.

Do I need to maintain a Business Associate Agreement (BAA) with both Rackspace and AWS?

For the AWS accounts that Rackspace supports, you only need to sign a BAA with Rackspace.

How can I get a copy of the Fanatical Support for AWS BAA?

Please get in touch with your Fanatical Support for AWS Technical Account Manager (TAM) or Rackspace Sales Representative who can get you a copy of the Fanatical Support for AWS BAA.

CHAPTER

TWELVE

PASSPORT

The Fanatical Support for AWS offering includes access to our service at the *Aviator service level*. This is the same capability that Rackers use to access your environment. Passport manages the provisioning of short-lived, access-limited, fully-audited bastion servers within your AWS account's VPC that can either be used directly or as a jump host for direct connectivity to other EC2 instances in the same VPC. Passport solves for both network connectivity and authentication into your environment.

Passport's primary concept is an **Access Request**. Each access request defines who is accessing your account, which specific EC2 instances they are accessing, which bastion instance is being used, the duration of the access request, and the reason for the access. Access requests default to expiring after 55 minutes (in order to optimize for the hourly billing of the bastion instances), but can be extended up to 11 hours and 55 minutes. A bastion instance will only ever be used by a single user, helping to ensure the integrity of the bastion operating system for each subsequent access request.

As an example, a Racker receiving a CloudWatch monitoring alarm for CPU utilization on your database server might create an access request referencing the alert ticket and granting them access to your active and passive database instances. Once troubleshooting and remediation is complete, the Racker completes the access request, immediately removing the bastion instance and all associated access.

All access request actions, from access request creation through expiration, are logged in *Logbook*.

12.1 Getting Started

To get started with Passport, follow these instructions:

1. Ensure that your AWS account is at the Aviator service level and that you are an Admin on the AWS account.
   - You can view your service level using the AWS Accounts list in the Fanatical Support for AWS Control Panel.
   - If needed, ask your Account Owner to visit the User Management page to provide Admin access to **Fanatical Support for AWS** on each AWS account for which you need access.
2. Install the ScaleFT *Server Agent* on your existing and/or new EC2 instances.
3. Install the ScaleFT *Workstation Tools* on your Mac OS X, Linux, or Windows workstation.
4. Follow the instructions in the *Workstation Tools* section to enroll your workstation.
5. Browse to the Passport section of the Fanatical Support for AWS Control Panel and click the **Create Access Request** button. Complete and submit the form.
6. Once your bastion instance is provisioned, click the links within the UI to access your servers. They will use a URL handler registered by the Workstation Tools to open a terminal window and execute the *sft ssh* or *sft rdp* command which will authenticate you (once per session), download the appropriate certificates in the background, and connect you to your desired EC2 instance.

Once you have completed your work, you can return to your access request in the Passport section of the Fanatical Support for AWS Control Panel and complete it to remove the bastion instance and other setup (such as security groups allowing access). As a reminder, your access request will automatically complete (expire) after 55 minutes unless you extend it via the same UI.

12.2 Overview

12.2.1 Network Connectivity

When initiating an access request, one of the required parameters is the source IP address that traffic will be initiated from inbound to the bastion instance. Rackers logging into your environment will actually proxy their traffic via the Rackspace shared support bastions before reaching their bastion instance in your VPC. Security groups are used to restrict inbound traffic to the bastion instance to SSH (TCP port 22) from the source IP specified in the access request.

Security groups are also used to allow traffic from the bastion inbound to your other EC2 instances. In an attempt to avoid the default AWS limit of five security groups per EC2 instance, a single security group is leveraged per EC2 instance regardless of the number of bastions active in your environment at any point in time. Each instance's bastion security group contains a list of all bastions that correspond to valid access requests for the instance in question. Once the final access request is completed or expires, the security group is automatically removed.

There are a couple of important considerations to account for in order to avoid interfering with the bastion service:

1. It is important to ensure that no more than four security groups per EC2 instance are used on your account in order to allow for a fifth security group to be provisioned on-demand for bastion access. If five security groups are already in use and you have the default AWS limits, the bastion access request could fail and delay Racker troubleshooting of issues impacting your environment.
2. Network ACLs (NACLs) have the potential to interfere with the specific traffic flows expected to be allowed by security groups, and could delay Racker troubleshooting of issues impacting your environment. Please *contact your Support team* prior to implementing NACLs so that we can help you avoid this issue.

12.2.2 Preferred Subnet

After creating an access request, Passport provisions a bastion for establishing secure connections to your EC2 instances. To provision the bastion, Passport needs a public subnet that allows traffic to the Internet Gateway for your VPC.

By default, Passport will try to find a suitable public subnet itself, which works for most customers. However, Passport may have difficulty finding a suitable public subnet in some environments, which may cause Passport errors.

If you have a particular subnet you want Passport to use, you can specify your preferred Passport subnet by simply creating the tag *passport=true* for the subnet. Passport can associate only one subnet with the bastion, so if a particular VPC has multiple subnets with the tag *passport=true*, Passport will try using the first subnet it finds. This may be undesirable, so we recommend that a VPC has only one preferred Passport subnet.

If you have no preferred Passport subnet, Passport falls back to automatically finding a default subnet. If you have subnets that you definitely do not want Passport to try using as the default subnet, you can create the tag *passport=false* for those subnets. This is valuable if you want to blacklist particular subnets from being used by Passport.

12.2.3 Authentication

Rackspace uses the ScaleFT Access offering from ScaleFT to generate temporary, short-lived (5 minute) authentication certificates for SSH and RDP that are signed by unique managed certificate authorities placed on each EC2 instance by the ScaleFT agent. The certificates are downloaded to a registered client workstation being used by an authenticated

user, and rotation is managed every 5 minutes while a valid, non-expired access request is present in the bastion service.

ScaleFT Access is used to provide authentication to both the bastion instance and other EC2 instances in your environment.

12.2.4 Tags

All bastion instances and security groups created by Passport are tagged with the following keys:

- `Application` (always has a value of "Rackspace Fanatical Support for AWS - Passport")
- `PassportRequestId`
- `PassportRequestedBy`
- `PassportRequestorType` (denotes whether the access request was created by a customer or a Racker)

12.3 ScaleFT Agents and Tools

12.3.1 Server Agent

The ScaleFT Server Agent must be installed on each of your servers that you wish for members of either the Rackspace team, your team, or both to have access to via Passport. The agent will automatically register with our centralized ScaleFT control plane. The easiest way to initiate the install is to include the following script in your EC2 instance's user data, which will automatically execute the script the first time the server boots. For existing servers, you can run the script at any time.

The agent will configure SSH (Linux) or RDP (Windows) to authenticate certificates against the Certificate Authority (CA) that the agent downloads (specific to your AWS account), but should not impact existing users that you already have in place.

Install and Configure

Ubuntu and Debian

1. Add the *ScaleFT package repository* for your Linux distribution
2. Execute the following commands:

```
#!/bin/bash
mkdir -p /etc/sft
# Set the Rackspace ScaleFT Control Plane URL
echo "InitialURL: https://scaleft.api.manage.rackspace.com" > /etc/sft/sftd.yaml
# Install the ScaleFT Server Tools
sudo apt-get install scaleft-server-tools
```

Red Hat, CentOS, and Fedora

1. Add the *ScaleFT package repository* for your Linux distribution
2. Execute the following commands:

```
#!/bin/bash
mkdir -p /etc/sft
# Set the Rackspace ScaleFT Control Plane URL
echo "InitialURL: https://scaleft.api.manage.rackspace.com" > /etc/sft/sftd.yaml
# Install the ScaleFT Server Tools
sudo yum install scaleft-server-tools
```

Windows Server

Supported Operating Systems: Windows Server 2012, Windows Server 2012 R2, Windows 2016 (except Nano Server, which doesn't support RDP)

1. Specify the following User Data (or remove the <powershell> tags and execute the commands directly via PowerShell):

```
<powershell>
# Set the Rackspace ScaleFT Control Plane URL
New-Item -Path 'C:\Windows\System32\config\systemprofile\AppData\Local\ScaleFT\sftd.yaml' -Value

# Install ScaleFT Server Tools
$installer_url = "https://dist.scaleft.com/server-tools/windows/latest/ScaleFT-Server-Tools-late
$installer_path = [System.IO.Path]::ChangeExtension([System.IO.Path]::GetTempFileName(), ".msi")
(New-Object System.Net.WebClient).DownloadFile($installer_url, $installer_path)
msiexec.exe /qb /I $installer_path
</powershell>
```

12.3.2 Workstation Tools

The ScaleFT Workstation Tools provide an easy way to manage the short-lived certificates that are issued by ScaleFT Access. Follow these instructions to initially install and configure the Workstation Tools.

The Workstation Tools will automatically register a URL handler which is used to provide convenient login links from the Access Request Details page in the Control Panel.

Install and Configure

Mac OS X

1. Download the latest Workstation Tools from https://dist.scaleft.com/client-tools/mac/latest/ScaleFT.pkg and run the installer.
2. Run the following command, making sure to replace **<your_rackspace_account_number>** with your six or seven-digit Rackspace account number (which can be found by clicking on the account dropdown in the top right of the Fanatical Support for AWS Control Panel):

```
sft enroll --default --url https://scaleft.api.manage.rackspace.com --team <your_rackspace_accou
```

3. Follow the on-screen prompts to login and complete the registration process.

Ubuntu and Debian

1. Add the *ScaleFT package repository* for your Linux distribution

2. Execute the following commands:

```
sudo apt-get install scaleft-client-tools
sudo apt-get install scaleft-url-handler
```

3. Run the following command, making sure to replace **<your_rackspace_account_number>** with your six or seven-digit Rackspace account number (which can be found by clicking on the account dropdown in the top right of the Fanatical Support for AWS Control Panel):

```
sft enroll --default --url https://scaleft.api.manage.rackspace.com --team <your_rackspace_accou
```

4. Follow the on-screen prompts to login and complete the registration process.

Red Hat, CentOS, and Fedora

1. Add the *ScaleFT package repository* for your Linux distribution
2. Execute the following commands:

```
sudo yum install scaleft-client-tools
sudo yum install scaleft-url-handler
```

3. Run the following command, making sure to replace **<your_rackspace_account_number>** with your six or seven-digit Rackspace account number (which can be found by clicking on the account dropdown in the top right of the Fanatical Support for AWS Control Panel):

```
sft enroll --default --url https://scaleft.api.manage.rackspace.com --team <your_rackspace_accou
```

4. Follow the on-screen prompts to login and complete the registration process

Windows

Supported Operating Systems: Windows 8, Windows 10

1. Download the ScaleFT installer and run the installation MSI.
2. Open a command prompt and run the following command, making sure to replace **<your_rackspace_account_number>** with your six or seven-digit Rackspace account number (which can be found by clicking on the account dropdown in the top right of the Fanatical Support for AWS Control Panel):

```
sft enroll --default --url https://scaleft.api.manage.rackspace.com --team <your_rackspace_accou
```

3. Follow the on-screen prompts to login and complete the registration process.
4. The first time ScaleFT is run on a Windows system it needs to be started manually from a command line:

```
%USERPROFILE%\AppData\Local\Apps\ScaleFT\ScaleFT.exe
```

   When the first Passport login request executes, you will be prompted to remember the association for access request links. Select "Yes".

   When ScaleFT is running you will see a white, 3-lobed icon in the system tray near the clock.

5. Your monitor resolution may require that you adjust display settings for ScaleFT. You can adjust your ScaleFT display size by setting a specific resolution, or starting in Fullscreen mode. See the following command examples:

Set screen resolution:

```
sft config rdp.screensize 1280x1024
```

Start in Fullscreen mode:

```
sft config rdp.fullscreen true
```

12.3.3 ScaleFT Package Repositories

ScaleFT distributes client and server packages for Linux via APT and RPM repositories.

Ubuntu and Debian

```
# Add the ScaleFT apt repo to your /etc/apt/sources.list system config file
echo "deb http://pkg.scaleft.com/deb linux main" | sudo tee -a /etc/apt/sources.list

# Trust the repository signing key
curl -C - https://dist.scaleft.com/pki/scaleft_deb_key.asc | sudo apt-key add -

# Retrieve information about new packages
sudo apt-get update
```

Red Hat, CentOS, and Fedora

```
# Add the ScaleFT yum repository
curl -C - https://pkg.scaleft.com/scaleft_yum.repo | sudo tee /etc/yum.repos.d/scaleft.repo

# Trust the repository signing key
sudo rpm --import https://dist.scaleft.com/pki/scaleft_rpm_key.asc
```

12.3.4 Known Issues and Suggestions

1. The following is a list of known issues or errors encountered by users:

- Passport does not support EC2 instances with multiple Elastic Network Interfaces (ENIs)
- ScaleFT requires manual start-up the first time it is run on a Windows workstation. See instructions earlier in this article under Workstation Tools.
- Client-side error for expired authentication token. Contact your Fanatical Support team for assistance.
- sshd refuses to authenticate your ScaleFT-issued key. Contact your Fanatical Support team for assistance.

2. Log files can be helpful when troubleshooting an issue. When contacting your Fanatical Support team for assistance, please attach your client logs to the support ticket.

Log files are typically stored in the following directories::

```
Windows: %USERPROFILE%\AppData\Local\ScaleFT\Logs
Linux: ~/.cache/ScaleFT/logs/sft/
Mac: ~/Library/Logs/ScaleFT/sft/
```

12.4 Advanced Usage

12.4.1 Copying files

The *sft* command line utility can be used via scp to copy files to a target instance:

```
# syntax
scp -o "ProxyCommand sft proxycommand <target_instance> --via <passport bastion>" <src> <target_insta

# example copying files.tar.gz to i-18a61fa5
scp -o "ProxyCommand sft proxycommand i-18a61fa5 --via i-8a982137" files.tar.gz i-18a61fa5:~/
```

<*target_instance*> can be a hostname or instance ID - see *sft list-servers* and *sft resolve* <*target_instance*>.

12.4.2 Forwarding ports

The *sft* command line utility can also be used to forward ports from a Passport bastion to other network addresses in AWS:

```
# syntax
sft ssh -L <local port>:<network address reachable by passport bastion>:<remote port> <passport basti

# example forwarding localhost:13306 to an RDS instance inside AWS on port 3306
sft ssh -L 13306:my_instance.us-east-1.rds.amazonaws.com:3306 i-0e41104c88525fbc5
```

Once the above *sft ssh* command is successfully running, you can use familiar local tools and connect them to *localhost:13306* to work with your RDS instance. Note that you must also have security group rules in place that permit access from your Passport bastion to the AWS resource you're accessing.

CHAPTER

THIRTEEN

LOGBOOK

AWS and Rackspace generate detailed control plane logs for all activities taking place in your Fanatical Support for AWS account(s). This data is aggregated from a number of different sources:

- AWS CloudTrail: detailed logs of all AWS API requests made on your account to supported AWS services
- Fanatical Support for AWS shared management system and user interfaces: view control panel logins and other actions (such as creating a new AWS account or modifying user permissions)
- Fanatical Support for AWS environment access: any time a Racker or one of your employees accesses your AWS environment by creating an access request and provisioning a temporary bastion, view the specific resources they had access to, the source of their access request, and other associated details throughout the duration of the access request

The section of the Fanatical Support for AWS Control Panel provides a timeline-based view of all of these activities. You can view the activities for a specific AWS account or view activities across all of your AWS accounts. You can also filter/facet the results to find the specific activities you are looking for. Logbook retains the last 90 days of historical data indexed for you to explore.

The information in Logbook can prove to be extremely valuable if you need to view the changes your employees, our Rackers, or automated processes made to your environment when troubleshooting an issue or reviewing the root cause of a service impacting event.

CHAPTER

FOURTEEN

COMPASS

is a set of tools that help you understand several dimensions of your AWS account(s). We perform an automated analysis of control plane (e.g. data visible through the AWS APIs) details to provide a thorough look at opportunities to improve the cost, reliability, redundancy, and security of your environment. We also present robust inventory information that provides an at-a-glance view of what resources you are running across all AWS public regions.

For Navigator accounts, our AWS experts are available to provide support and guidance on Compass recommendations. For Aviator accounts, our Fanatical Support for AWS support teams will work with you to proactively execute on Compass recommendations to help reduce AWS spend, improve your security posture, and implement other optimizations.

You can access Compass by clicking the Compass link in the Fanatical Support for AWS Control Panel.

14.1 Permissions

Compass provides views of individual AWS accounts for which you have AWS control plane access. If you have access to all AWS accounts on your Rackspace account you will also be able to view a multi-account view that includes aggregated data about all of your AWS accounts.

CHAPTER

FIFTEEN

WAYPOINT

Waypoint is a tool that provides high-level, concise information about costs, risks, and other key operational information associated with your AWS accounts. By querying information from various APIs from both AWS and Rackspace, it provides a consolidated view of key information. The goal of Waypoint is to keep you informed about what's happening in your AWS environments and to ensure we work together on improving your experience.

Waypoint is a near-real-time dashboard for the current month, and it also provides historical reports for the previous 12 months, giving you both current and trending information.

For the current month, cost data is updated approximately once per day. Typically, by the tenth day of a month, we finalize the cost data for the previous month (though the process may take a few days longer to run).

Users with access to all AWS accounts within a Rackspace account can access Waypoint by clicking the Waypoint link in the Fanatical Support for AWS Control Panel. Waypoint is available only to users with access to all AWS accounts because it summarizes details for all those accounts.

Users with access to only a subset of all AWS accounts within a Rackspace account can instead access Usage, which provides billing details for the subset of accounts. To access Usage, click the Usage link in the Fanatical Support for AWS Control Panel.

Users will see either Waypoint (for those with access to all AWS accounts) or Usage (for those with access to only some AWS accounts), but not both.

CHAPTER

SIXTEEN

WATCHMAN

AWS CloudWatch is the primary monitoring system used by our Fanatical Support for AWS support teams. CloudWatch provides a wide variety of metrics that cover the entire suite of AWS services - from CPU utilization and disk I/O on EC2 instances to network throughput of your ELB load balancers.

While AWS CloudWatch is available to Fanatical Support for AWS accounts at all service levels, customers using our Aviator service level can opt to have a Racker respond to unexpected deviations in metrics. Watchman is the system responsible for receiving CloudWatch alarms and creating tickets on your Rackspace account.

16.1 CloudWatch Alarms

CloudWatch Alarms can be triggered to fire when the value of a CloudWatch metric deviates from its expected value. For example, if CPU utilization on an EC2 instance exceeds 80% for a period of five minutes or greater, the CloudWatch alarm can be configured to send an alert to a Rackspace-managed SNS (Simple Notification Service) topic (named *rackspace-support*) that will generate a ticket for further investigation by a Racker.

The *rackspace-support* SNS topic is configured in each region when your AWS account is first setup for Fanatical Support for AWS. A subscription to the SNS topic is created for a centralized region-specific SQS (Simple Queue Service) queue that resides in our shared management services account. Our shared management services system continually monitors these queues and generates a ticket when a valid CloudWatch alarm is received from an Aviator service level account.

Note: While the SNS topic described above is present on every Fanatical Support for AWS account, only accounts at the Aviator service level will have tickets generated. If your account is at the Navigator service level, no action will be taken for CloudWatch alarm notifications sent to your account's *rackspace-support* SNS queue.

16.2 Custom CloudWatch Configuration

CloudWatch allows for the creation of custom metrics to allow monitoring the things that are most critical to the uptime of your applications. As an Aviator customer, you can create custom CloudWatch metrics and alarms, as well as send notifications to the *rackspace-support* SNS topic if you desire a Racker response to triggered alarms. We do recommend that you work with a Racker when first creating custom CloudWatch metrics and alarms so that we can ensure that everything is configured properly and that the desired Racker response is clearly documented.

CHAPTER

SEVENTEEN

SUPPORT

There are multiple ways to receive Fanatical Support for your AWS account(s). A helpful Racker is always just a phone call or ticket away. We are available live 24x7x365.

17.1 Tickets

One of the primary ways that you can interact with a Racker is by creating a ticket in the Fanatical Support for AWS Control Panel. Once logged in, click the Support button in the black bar at the top of the screen and follow the links to create a new ticket or view an existing ticket.

Our automated systems will create tickets for events on your AWS account(s) that require either your attention or the attention of a Racker. For example, our *Rackspace Watchman* will create a ticket when an alarm is raised that requires attention.

Any time a ticket is updated, you will receive an email directing you back to the Control Panel to view the latest comments.

17.2 Phone

Would you prefer to speak to a live Racker? Give our team a call at 877-417-4274 (US) or 0800-033-4045 (UK) and we'll be happy to assist you. Additional international contact numbers are available on our Contact Us page.

CHAPTER

EIGHTEEN

AWS MARKETPLACE

You are able to purchase items from the AWS Marketplace for use in your AWS account. Note that any purchases from the AWS Marketplace will be billed to you along with your AWS infrastructure and Rackspace management fees on your monthly invoice. Additionally, any purchases from the AWS Marketplace will calculated as AWS infrastructure for purposes of calculating Rackspace management fees.

18.1 Legal Terms

We may agree to install third party software (for example, from an AWS marketplace) as part of the Services. Where such activity requires the acceptance of an End User License Agreement (or similar terms), you hereby authorize Rackspace to accept such terms on your behalf, agree to be bound by and adhere to such terms, and acknowledge that you, and not Rackspace are bound by such terms. We will notify you via ticket when we accept such terms on your behalf and direct you to a copy.

Caution: Information in this section refers to a future offering that may not be available at this time. This documentation may also be updated at any time. **Please reach out to your Account Manager for more information.**

CHAPTER

NINETEEN

INFRASTRUCTURE AS CODE (BETA)

AWS environments built under the Fanatical Support for AWS Aviator service level are managed through a process known as Infrastructure as Code (IaC), specifically using an industry-standard tool called Terraform. This means that changes to your environment are managed through Terraform, and not through the AWS Console directly. Changes in the AWS Console can conflict with Terraform management, resulting in downtime, data loss, or delays to reconcile these manual changes. It is important that all changes to your environment are managed with Terraform. If you need assistance applying a change, the Rackspace Support team will be happy to help.

Caution: Information in this section refers to a future offering that may not be available at this time. This documentation may also be updated at any time. **Please reach out to your Account Manager for more information.**

19.1 Using GitHub

There are three kinds of repositories in the rackspace-infrastructure-automation organization:

- Rackspace-wide or Public Projects

  Rackspace-wide projects containing documentation and/or reusable code to build Customer infrastructure (Documentation, Terraform modules, etc)

  Naming convention: *<Cloud Platform or rackspace>-<Tool/Purpose>*

  Examples: rackspace-documentation, aws-terraform-base_network

- Customer shared repository

  Customer repositories that are shared across all of a single Customer's accounts (Reusable Terraform artifacts)

  Naming convention: *<Customer number>-<cloud provider>-<Human Readable, arbitrary name>*

  Examples: 12345-aws-Customer1, 67890-aws-Customer2, etc|

- Customer account-level repository

  Customer repositories specific to a cloud account (code used to directly build AWS, GCP, Azure, etc account resources)

  Naming convention: *<Customer number>-<cloud provider>-<cloud provider account id>-<Human Readable, arbitrary name>*

  Examples: 12345-aws-45378692529-Customer1, 67890-aws-9085432530-Customer2, etc|

19.1.1 Branches & forks

- All repositories should have a README.md committed to *master*
- *master* branch is protected, only controlled by CI/CD. *master* should always reflect a sane, valid, current state
- Short-lived branches will be used for changes
- Forks won't build using our CI/CD system, by design
- Releases will exist as a tag, and will be created using the GitHub releases mechanism.

19.1.2 Membership and collaboration

- Racker accounts will be added & removed to the Rackspace team, automatically.
- Rackspace will automate adding and removing members & outside collaborators. We anticipate additional work around Identity and Identity federation in the future.
- Rackers will be added to specific teams based on which level of access and which cloud providers they work with.
- Customers must request GitHub users be added or removed from their repositories, and will only be serviced during US or UK Rackspace business hours.

Caution: Information in this section refers to a future offering that may not be available at this time. This documentation may also be updated at any time. **Please reach out to your Account Manager for more information.**

19.2 Terraform Standards

19.2.1 Environments

We define an 'environment' as similar deployments of AWS resources, either in separate accounts or a single account.

Generally, any Terraform functionality used in multiple environments should be placed in the shared modules repository. There should not be any environment-specific state in the shared modules repository.

We shouldn't be running sub-directories' TF files just to reduce the blast radius of a change. Use plan, consider moving to separate accounts, or something else that doesn't rely on obfuscation.

19.2.2 Running terraform

- All changes should be made through CI/CD tooling, and Terraform should not be run locally – especially *terraform apply*
- Terraform versions and provider versions should be pinned, as it's not possible to safely downgrade a state file once it has been used with a newer version of Terraform
- Create "GitHub release" objects for releases, which automatically make tags, lets us define release notes / change log, and provides download links
- S3 (standard region) with DynamoDB locking, with versioning & server-side encryption in S3
- Standard bucket name: *tf-state-(aws acct #)*
- All changes should be mapped to a specific AWS account as a GitHub repo

- Read only state should be visible directly in S3

19.2.3 Module and general Terraform authoring best practices

- use semantic versioning for shared code and modules
- always point to GitHub releases (over a binary or master) when referencing external modules
- always extend, don't re-create resources manually
- parameters control counts, for non-standard numbers of subnets/AZs/etc.
- use overrides to implement Rackspace best practices
- use variables with good defaults for things Rackspace expects to configure

19.2.4 Organization of Terraform files (generic guidelines)

The root directory, all empty directories, and all reusable modules should have a *README.md*. Any other docs may to go in a *docs/* directory. Any unsupported/Customer changes that aren't covered under support should be noted in the README.md. Each terraform module will be in a separate repository, dropping the *Rackspace* from the repo naming scheme

19.2.5 Repository Structure

- Modules should use semantic versioning light (Major.minor.0) for AWS account repositories
- Standard *main.tf, variables.tf, output.tf* structure
- *auto.tfvars* should be present and used in CI testing

Account:

main.tf should always be present in the root of the repository, and should define aws provider and pin version of provider

If Customer uses multiple regions in a single AWS account, directories for each region should be created as well. Each regional directory should have a *main.tf* with a region set on the AWS provider.

```
./main.tf (aws provider, pin version)
./us-east-1/main.tf (aws provider with region set)
./us-west-2/main.tf (aws provider with region set)
```

If a repository only deploys to a single region in this AWS account, the regional directories may be omitted.

19.2.6 File Naming Conventions

```
terraform.tfvars (in .gitignore)
terraform.tfvars.rackspace (in .gitignore, our overrides for our own local development)
terraform.tfvars.dist (variables for whatever the actual environment it's going to, will be fed to Te
```

Rackspace recommends using tfvars files for reusable modules where variables will usually be overrode, not necessarily for Customer's AWS account-specific modules.

19.2.7 Custom Modules

When to make a custom Terraform module (vs. using existing aws provider primitives)

Prefer to use built-in primitives (i.e. resources) unless there's a strong reason (see below) to build our own modules.

- When multiple resources should always be used together (e.g. a CloudWatch Alarm and EC2 instance, for autorecovery)
- When Rackspace has a strong opinion that overrides default values for a resource
- When module re-use remains shallow (don't nest modules if at all possible)

Strive to use modules (prefer Rackspace modules, then built-in aws provider resources) 100% of the time, and override where necessary.

19.2.8 Style Guide

- Follow syntax at https://github.com/hashicorp/hcl/blob/master/README.md
- Format/Lint: terraform fmt for linting and formatting - https://www.terraform.io/docs/commands/fmt.html
- Validate syntax: terraform validate - syntax check
- snake_case for all resources in TF
- 2 space indents - no hard tabs (from standards above)
- All variables declared in variables.tf, not main.tf
- Always import specific tags, never master or commit hashes, when referring to external modules
- Never hard-code regions in modules, only in account-specific repositories as a variable - Ideally use provider definitions to set region
- Use module-relative paths and file helper when you need user-data in a module
- Favor separate resources over inline blocks

Caution: Information in this section refers to a future offering that may not be available at this time. This documentation may also be updated at any time. **Please reach out to your Account Manager for more information.**

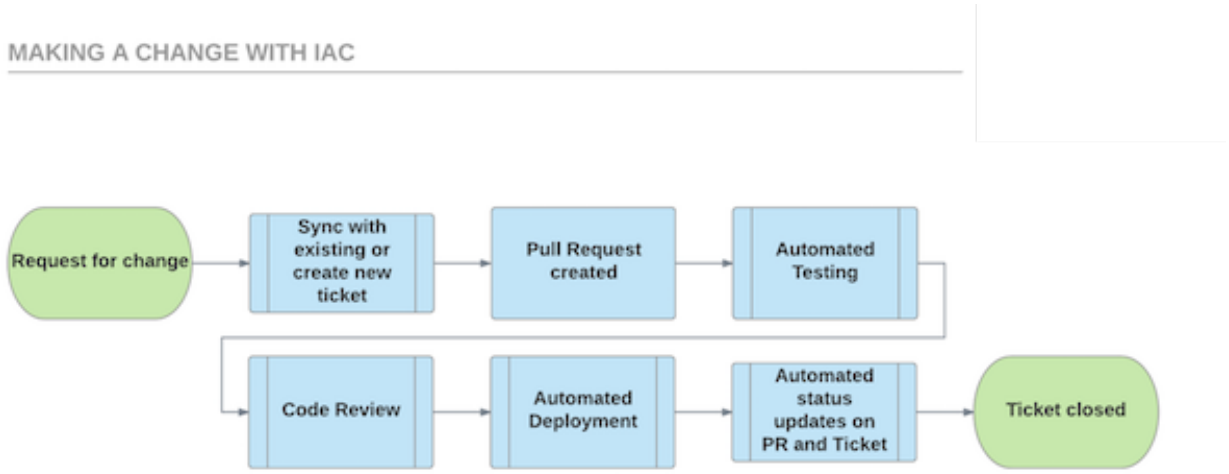
19.3 Making Changes

19.3.1 Glossary

- PR: Pull request in GitHub
- Racker QC (technical): Does this change do what the Customer asked, in an appropriate and well-designed way?
- Approval (non-technical): Can Rackspace proceed with making the change (now or at a specific time)?
- Implicit Approval: Time-boxed, but Customer could explicitly hit approve or deny to speed up the process.
- Explicit Approval : Customer must respond; Rackspace doesn't proceed without approval. For any Customer approval, Customer is responsible for any implied QC step.
- Automated testing: Are the code changes technically correct? (Runs all on all types of changes.)

Rackspace recommends specific best practices for what types of approval and QC steps should happen for a change (defaults). Customers may override what approval is required for a specific change by request.

19.3.2 Change workflows



Customers may initiate changes via pull request or ticket. Detailed conversation happens in GitHub; customers must have a GitHub account if they would like to participate in development. Major events in this workflow are reflected in the companion ticket. Racker & Customer approval is required at both code review and deployment steps (with exceptions for standard changes or by customer request). Changes will always require a reviewer that is different than the change author. Here some specific personas (user types) and the workflows they map to. Note that these are really all the same workflow, but have different levels of approval (explicit vs. implicit).

Passenger

Request is approval, unless otherwise requested. Rackers write & QC all changes.

- Customer initiated ticket: Customer's ticket gives approval, 1 Racker writes, 1 Racker QCs.
- Rackspace initiated PR: Customer must approve it (implicit), 1 Racker writes, 1 Racker QCs.

Conversation happens in ticket; Customers not expected to be able to approve via GitHub.

Trainee

Approval is implicit (& may include Customer QC), unless otherwise requested. Racker or Customer (or both) makes changes, Racker QCs all changes.

- Customer initiated ticket: Customer's ticket gives approval, 1 Racker writes, 1 Racker QCs.
- Customer initiated PR: Racker or Customer may add additional commits, 1 Racker QCs. If Racker makes changes, Customer approval is implicit.
- Rackspace initiated PR: Customer must approve it (implicit), 1 Racker writes, 1 Racker QCs.

Conversation happens in GitHub; Customers are expected to have a GitHub account. Major PR state changes reflect in companion ticket.

Copilot

Approval is explicit (& may include Customer QC), unless otherwise requested. Racker or Customer (or both) makes changes, Racker QCs all changes.

- Customer initiated ticket: Customer's ticket gives approval, 1 Racker writes, 1 Racker QCs.

- Customer initiated PR: Racker or Customer may add additional commits, 1 Racker QCs. If Racker makes changes, Customer must approve it (explicit).
- Rackspace initiated PR: Customer must approve it (explicit), 1 Racker writes, 1 Racker QCs.

Conversation happens in GitHub; Customers are required to have a GitHub account. Major PR state changes reflect in companion ticket.

We should provide a mechanism for explicit Customer approval (entered by Racker or Customer), regardless of which UI (GitHub/Rackspace Ticketing/Rackspace Control Panel). We will always document exactly how this happened. We may eventually build tooling for Customer initiated PRs to be done automatically too. We need to study this workflow further to identify what approvals may be necessary in this scenario.

19.3.3 Deployment process

- One repository/AWS account at a time. We lock on the repo/AWS/GCP/cloud account level, and may deploy to multiple regions in that one file.
- When Rackers have multiple changes staged at once, they may encounter a need to re-plan changes during a maintenance window, no matter how proactive Rackers want to be. We must consider this extra time when scheduling a maintenance.
- Until unlimited availability, Rackspace will not automatically deploy changes without Racker interaction or Racker QC, even if two Customer users peer-review.
- Until unlimited availability, Rackspace doesn't have a default list of changes that can bypass QC.
- Failed changes fall into two categories: 1. something didn't get applied (e.g. couldn't build an instance in us-east-1). Needs workflow. 2. something was applied, but didn't have the desired effect (e.g. opened the wrong port in a security group). Use existing workflow.

The relationship between Rackspace Ticketing and GitHub is still to be determined; Rackspace doesn't think every single GitHub event should be copied, but high level details may be appropriate.

When opening a ticket, Rackspace needs clarification on if the Customer does or does not want to be in the approval, code review, or QC process. Rackspace should have optional blocks for these steps in the process in case the Customer wants to perform any of these. These should be default behaviors (Customer does code review), but there should be an override. See change notes above.

Please be sure to inform Rackspace of any specific change window or maintenance window requirement you may have.

To raise issues, questions, and changes that aren't already represented as pull requests, Customers should open a Rackspace ticket. GitHub's Issues feature is disabled on all Rackspace-managed repositories.

Caution: Information in this section refers to a future offering that may not be available at this time. This documentation may also be updated at any time. **Please reach out to your Account Manager for more information.**

19.4 Deploying Code on AWS

From time to time, you will be faced with a decision about how to deploy a file, instance configuration, agents, or even entire applications into an AWS environment (we refer to these collectively as 'artifacts' below). This page is intended to be our Fanatical AWS "best practices" and opinions around each option.

Many of these options provide an initial bootstrapping step that can then be used to perform additional work or copy additional files from other locations.

19.4.1 Bake files into AMI

Copy needed artifacts to an EC2 image, create an Amazon machine image (AMI), and rotate instances using the new AMI.

Variations:

- Create an instance manually and create an AMI
- Use tools like Packer to automatically build AMIs

Benefits:

- No dependence on external sources
- Fastest instance boot time
- Guaranteed state
- Strongly version controlled
- Can release updates to multiple files simultaneously
- Allows for testing before deployment
- Encryption possible

Drawbacks:

- Updates require a new AMI be built
- Storage cost of AMIs
- May have many AMIs with duplicate artifacts (e.g. one AMI per region)
- Requires additional management to clean up old AMIs
- Some artifacts need to be staged before AMI taken (i.e. sysprep style)
- Some artifacts (e.g. agents) may conflict with user-data/cloud-init

19.4.2 AWS CodeDeploy

Install CodeDeploy agent using one of the methods in this table, then use APIs to drive deployment from GitHub, S3, or another supported Git hosting provider. Embed additional artifacts in the deployment artifact or fetch them during CodeDeploy lifecycle hook execution.

Variations:

- Store artifacts in git and deploy them as part of application
- Fetch artifacts in S3 or another location

Benefits:

- CodeDeploy offers blue-green and canary deployment options
- Allows for live deployments of configuration files without rotating AMIs
- Config files can be tied to application deployments
- Encryption possible with S3

Drawbacks:

- Another agent to maintain and update

- Secrets should not be stored in git repositories
- Requires a deploy of application to update configuration files

19.4.3 User Data

Embed scripts in user-data or in cloud-init (called by user data). These scripts can fetch artifacts from S3, embed smaller artifacts directly, or even add package repositories and install packages.

Variations:

- Fetch files in S3 on boot
- Write files directly on boot

Benefits:

- Fastest ability to update files (just build more instances)
- Ability to always download the latest versions (e.g. for agents)
- Customer can deploy secrets without Rackspace needing to be involved
- S3 can be secured and accessible only from a VPC Endpoint
- Encryption possible

Drawbacks:

- Slows down provisioning of new instances
- Dependence on external source to boot instance
- Difficult / slow to debug, as autoscaling may terminate unhealthy instances
- No guarantee that instances will have same artifact versions
- Security controls must be managed appropriately, with 3rd party code without review on new versions

19.4.4 Native AWS APIs

Write your application to utilize native EC2 SSM Parameter Store or other AWS storage services to directly retrieve artifacts at runtime.

Variations:

- EC2 SSM Parameter Store
- S3, DynamoDB, etc

Benefits:

- No need to manage config file deployment
- Applications can be dynamically updated

Drawbacks:

- Significantly more opaque than a file-based method
- Requires a very good knowledge of the AWS APIs
- May require a re-architecture of the application

19.4.5 Terraform Module

Terraform offers a template provider that can be used to embed artifacts inline or as separate files. You can then use these artifacts via data sources when building a cloud-init configuration.

Variations:

- Keep files in git
- Embed scripts directly in Terraform file

Benefits:

- Terraform native functionality to deploy files with dynamic values
- Allows for collaboration on artifacts in Git
- Uses cloud-init for cross-platform functionality
- Less files strewn around various places
- Easier to pull in dynamic values from other APIs
- Single place to manage instance configuration

Drawbacks:

- Requires co-locating configuration files and Terraform files
- Inline artifacts make Terraform harder to read; painful escaping of strings
- Not in the spirit of Terraform or Infrastructure as Code
- Cannot dynamically update artifacts without applying Terraform again

> **Caution:** Information in this section refers to a future offering that may not be available at this time. This documentation may also be updated at any time. **Please reach out to your Account Manager for more information.**

19.5 Secrets management

19.5.1 In Terraform

Rackspace recommends storing secrets for Terraform using AWS KMS; embed ciphertext values as data sources in Terraform configurations. Here's some of the specifics and considerations:

- Use *aws_kms_key* to create a KMS key for use; you should apply a key policy that allows IAM roles and users to use the key, because federated accounts won't (e.g. most Rackers and Customers).
- You will need to manually use the AWS CLI (and the key-id for the key you created earlier) to encrypt your secrets (mind any line endings if you use *file://* to encrypt):

```
$ aws kms encrypt \
    --key-id 438290482-e36a-4803-a7d0-db436278 \
    --plaintext "super_secret" \
    --encryption-context resource=my_database,key=password \
    --output text --query CiphertextBlob
```

- Equipped with the ciphertext from the previous command, you can now use aws_kms_secret to expose the secret as a data source for further use in Terraform:

```
data "aws_kms_secret" "my_database" {
  secret {
    name    = "password"
    payload = "ciphertext"

    context {
      resource = "my_database"
      key      = "password"
    }
  }
}
```

19.5.2 In Packer

Rackspace recommends storing secrets for Packer using SSM Parameter Store. Both Rackspace and AWS have published blog posts with additional examples. Here's some of the specifics and considerations:

- Similar to Terraform above, you will need a KMS key and key policy. For accessing secrets during the Packer build process, you will also need to use an instance role that has permission *ssm:DescribeParameters* for * and *ssm:GetParameters* for the specific parameters you want to access.

Example of encryption command:

```
$ aws ssm put-parameter --name APPLICATION_SECRET_API_KEY \
                        --description "The encrypted secret API key for our awesome application" \
                        --value "super_secret" \
                        --type SecureString \
                        --key-id 438290482-e36a-4803-a7d0-db436278
```

- Knowing the parameter name and having a role that can access it, you are now able to retrieve it in EC2 instances or Packer provisioner scripts (or in your own application):

```
$ aws ssm get-parameters --names APPLICATION_SECRET_API_KEY --with-decryption
```

Caution: Information in this section refers to a future offering that may not be available at this time. This documentation may also be updated at any time. **Please reach out to your Account Manager for more information.**

19.6 Frequently Asked Questions

- Why am I seeing Terraform plans with EC2 instances that have one security group removed?

This is most likely from Passport. You should close the Passport request before making infrastructure changes, or restore the removed security group once the change has been applied.