



# INSTALLATION GUIDE

A guide for installing and upgrading CircleCI Server on AWS.

Docs Team

Version 2.17.3, 09/11/2019

<b>Installation Overview</b>	<b>1</b>
Support Packages	1
<b>System Requirements</b>	<b>2</b>
Services Machine	2
Nomad Clients	2
Server Ports	2
<b>Installation Prerequisites</b>	<b>8</b>
Private Subnet Requirements	8
Planning	8
<b>Installation on AWS with Terraform</b>	<b>11</b>
Define Variables for Terraform	11
Provision Instances	13
Access Your Installation	14
Installation Setup	17
Validate Your Installation	22
<b>Teardown</b>	<b>24</b>
<b>Upgrading a Server Installation</b>	<b>25</b>
Org Rename Script	25
Upgrade Steps Overview	29

# Installation Overview



CircleCI Server v2.17.x uses the CircleCI 2.0 architecture.

The following sections provide planning information, system requirements and step-by-step instructions for installing CircleCI Server on Amazon Web Services (AWS) with Terraform.

Refer to the [changelog](#) for what's new and fixed in this release.

If you are looking to update an existing installation, see our guide to [Upgrading a Server Installation](#).

## Support Packages

CircleCI 2.0 may be installed without a support package, on AWS, using the examples and instructions in this document. Alternatively, if you do decide to go ahead with a support package, there are a number of benefits, as detailed below:

### Non-AWS Platform Support

With a Platinum CircleCI support package it is possible to install and configure CircleCI on Azure or any other platform used in your organization. Contact [CircleCI support](#) or your account representative to get started.

### Externalization

With a Platinum support agreement, it is possible to improve performance and resilience by configuring the following services to run externally to the Services machine:

- PostgreSQL
- MongoDB
- Vault
- Rabbitmq
- Redis
- Nomad

Contact [CircleCI support](#) or your account representative to evaluate your installation against the current requirements for running external services.

# System Requirements

This section defines the system and port access requirements for installing CircleCI v2.17.

## Services Machine

The Services machine hosts the core of our Server product, including the user-facing website, API engine, datastores, and Nomad job scheduler. It is best practice to use an isolated machine.

The following table defines the Services machine CPU, RAM, and disk space requirements:

Number of daily active CircleCI users	CPU	RAM	Disk space	NIC speed
<50	8 cores	32GB	100GB	1Gbps
50-250	12 cores	64GB	200GB	1Gbps
251-1000	16 cores	128GB	500GB	10Gbps
1001-5000	20 cores	256GB	1TB	10Gbps
5000+	24 cores	512GB	2TB	10Gbps

## Nomad Clients

Nomad client machines run the CircleCI jobs that are scheduled by the Nomad Server on the Services machine. Following are the Minimum CPU, RAM, and disk space requirements per client:

- CPU: 4 cores
- RAM: 32GB
- Disk space: 100GB
- NIC speed: 1Gbps

The following table defines the number of Nomad clients to make available as a best practice. Scale up and down according to demand on your system:

Number of daily active CircleCI users	Number of Nomad client machines
<50	1-5
50-250	5-10
250-1000	10-15
5000+	15+

## Server Ports

Bellow all ports required by a CircleCI 2.0 installation are listed for each machine type.

## Services Machine

Port number	Protocol	Direction	Source / destination	Use	Notes
80	TCP	Inbound	End users	HTTP web app traffic	
443	TCP	Inbound	End users	HTTPS web app traffic	
7171	TCP	Inbound	End users	Artifacts access	
8081	TCP	Inbound	End users	Artifacts access	
22	TCP	Inbound	Administrators	SSH	
8800	TCP	Inbound	Administrators	Admin console	
8125	UDP	Inbound	Nomad Clients	Metrics	
8125	UDP	Inbound	Nomad Servers	Metrics	Only if using externalized Nomad Servers
8125	UDP	Inbound	All Database Servers	Metrics	Only if using externalised databases
4647	TCP	Bi-directional	Nomad Clients	Internal communication	
8585	TCP	Bi-directional	Nomad Clients	Internal communication	
7171	TCP	Bi-directional	Nomad Clients	Internal communication	
3001	TCP	Bi-directional	Nomad Clients	Internal communication	
80	TCP	Bi-directional	GitHub Enterprise / GitHub.com (whichever applies)	Webhooks / API access	
443	TCP	Bi-directional	GitHub Enterprise / GitHub.com (whichever applies)	Webhooks / API access	
80	TCP	Outbound	AWS API endpoints	API access	Only if running on AWS

Port number	Protocol	Direction	Source / destination	Use	Notes
443	TCP	Outbound	AWS API endpoints	API access	Only if running on AWS
5432	TCP	Outbound	PostgreSQL Servers	PostgreSQL database connection	Only if using externalised databases. Port is user-defined, assuming the default PostgreSQL port.
27017	TCP	Outbound	MongoDB Servers	MongoDB database connection	Only if using externalized databases. Port is user-defined, assuming the default MongoDB port.
5672	TCP	Outbound	RabbitMQ Servers	RabbitMQ connection	Only if using externalized RabbitMQ
6379	TCP	Outbound	Redis Servers	Redis connection	Only if using externalized Redis
4647	TCP	Outbound	Nomad Servers	Nomad Server connection	Only if using externalized Nomad Servers
443	TCP	Outbound	CloudWatch Endpoints	Metrics	Only if using AWS CloudWatch

## Nomad Clients

Port number	Protocol	Direction	Source / destination	Use	Notes
64535-65535	TCP	Inbound	End users	SSH into builds feature	
80	TCP	Inbound	Administrators	CircleCI Admin API access	
443	TCP	Inbound	Administrators	CircleCI Admin API access	
22	TCP	Inbound	Administrators	SSH	
22	TCP	Outbound	GitHub Enterprise / GitHub.com (whichever applies)	Download Code From Github.	
4647	TCP	Bi-directional	Services Machine	Internal communication	
8585	TCP	Bi-directional	Services Machine	Internal communication	
7171	TCP	Bi-directional	Services Machine	Internal communication	
3001	TCP	Bi-directional	Services Machine	Internal communication	
443	TCP	Outbound	Cloud Storage Provider	Artifacts storage	Only if using external artifacts storage
53	UDP	Outbound	Internal DNS Server	DNS resolution	This is to make sure that your jobs can resolve all DNS names that are needed for their correct operation.

## GitHub Enterprise / GitHub.com

Port number	Protocol	Direction	Source / destination	Use	Notes
22	TCP	Inbound	Services Machine	Git access	
22	TCP	Inbound	Nomad Clients	Git access	
80	TCP	Inbound	Nomad Clients	API access	
443	TCP	Inbound	Nomad Clients	API access	
80	TCP	Bi-directional	Services Machine	Webhooks / API access	

## PostgreSQL Servers

Port number	Protocol	Direction	Source / destination	Use	Notes
5432	TCP	Bi-directional	PostgreSQL Servers	PostgreSQL replication	Only if using externalized databases. Port is user-defined, assuming the default PostgreSQL port.

## MongoDB Servers

Port number	Protocol	Direction	Source / destination	Use	Notes
27017	TCP	Bi-directional	MongoDB Servers	MongoDB replication	Only if using externalized databases. Port is user-defined, assuming the default MongoDB port.



## RabbitMQ Servers

Port number	Protocol	Direction	Source / destination	Use	Notes
5672	TCP	Inbound	Services Machine	RabbitMQ connection	Only if using externalized RabbitMQ
5672	TCP	Bi-directional	RabbitMQ Servers	RabbitMQ mirroring	Only if using externalized RabbitMQ

## Redis Servers

Port number	Protocol	Direction	Source / destination	Use	Notes
6379	TCP	Inbound	Services Machine	Redis connection	Only if using externalized Redis
6379	TCP	Bi-directional	Redis Servers	Redis replication	Only if using externalized Redis, and using Redis replication (optional)

## Nomad Servers

Port number	Protocol	Direction	Source / destination	Use	Notes
4646	TCP	Inbound	Services Machine	Nomad Server connection	Only if using externalized Nomad Servers
4647	TCP	Inbound	Services Machine	Nomad Server connection	Only if using externalized Nomad Servers
4648	TCP	Bi-directional	Nomad Servers	Nomad Servers internal communication	Only if using externalized Nomad Servers

# Installation Prerequisites

CircleCI uses Terraform to automate parts of the infrastructure for your CircleCI Server install, so you will need to install this first:

- Visit [Download Terraform](#) and choose the correct package for your architecture.

Ensure you have the following information available before beginning the installation procedure:

- A CircleCI License file (`.r1i`). Contact [CircleCI support](#) for a license and request a cluster-enabled license to run jobs on dedicated instances for best performance.
- Your AWS Access Key ID and Secret Access Key.
- Name of your [AWS EC2 key pair](#).
- [AWS Region](#), for example `us-west-2`.
- AWS Virtual Private Cloud (VPC) ID and AWS Subnet ID. If your account is configured to use a default VPC, your default VPC ID is listed under Account Attributes, which you will find from the AWS management console on the EC2 dashboard page.
- Set your VPC (`enableDnsSupport`) setting to `true` to ensure that queries to the Amazon provided DNS server at the 169.254.169.253 IP address, or the reserved IP address at the base of the VPC IPv4 network range plus two will succeed. See the [Using DNS with Your VPC](#) Amazon Web Services documentation for additional details.

## Private Subnet Requirements

The following additional settings are required to support using private subnets on AWS with CircleCI:

- The private subnet for builder boxes must be configured with a [NAT gateway](#) or an [internet gateway](#) configured for the outbound traffic to the internet via attached route tables.



The subnet should be large enough to **never** exhaust the addresses.

- The [VPC Endpoint for S3](#) should be enabled. Enabling the VPC endpoint for S3 should significantly improve S3 operations for CircleCI and other nodes within your subnet.
- Adequately power the NAT instance for heavy network operations. Depending on the specifics of your deployment, it is possible for NAT instances to become constrained by highly parallel builds using Docker and external network resources. A NAT that is inadequate could cause slowness in network and cache operations.
- If you are integrating with [github.com](#), ensure that your network access control list (ACL) whitelists ports 80 and 443 for GitHub webhooks. When integrating with GitHub, either set up CircleCI in a public subnet, or set up a public load balancer to forward github.com traffic.
- See the [Services Machine](#) section of our overview for more information on the specific ports that need to be accessible to instances in your CircleCI installation.

## Planning

Have available the following information and policies before starting the installation:

- If you use network proxies, contact your Account team before beginning your install.
- Plan to provision at least two AWS instances, one for Services and one for your first set of Nomad Clients. Best practice is to use an `m4.2xlarge` instance with 8 vCPUs and 32GB RAM for both the Services and Nomad Clients instances.
- AWS instances must have outbound access to pull Docker containers and to verify your license. If you don't want to give open outbound access, see our [list of ports](#) that will need access.
- In order to provision required AWS entities with Terraform you will require an IAM User with the following permissions (See the [AWS guidance](#) on creating IAM users):

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:*"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3:::circleci-*",
        "arn:aws:s3:::circleci-*/*",
        "arn:aws:s3:::*"
      ]
    },
    {
      "Action": [
        "autoscaling:*",
        "sqs:*",
        "iam:*",
        "ec2:StartInstances",
        "ec2:RunInstances",
        "ec2:TerminateInstances",
        "ec2:Describe*",
        "ec2:CreateTags",
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateSecurityGroup",
        "ec2>DeleteSecurityGroup",
        "ec2:DescribeInstanceAttribute",
        "ec2:DescribeInstanceStatus",
        "ec2:DescribeInstances",
        "ec2:DescribeNetworkAcls",
```

```

        "ec2:DescribeSecurityGroups",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:ModifyInstanceAttribute",
        "ec2:ModifyNetworkInterfaceAttribute",
        "cloudwatch:*",
        "autoscaling:DescribeAutoScalingGroups",
        "iam:GetUser"
    ],
    "Resource": [
        "*"
    ],
    "Effect": "Allow"
}
]
}

```

# Installation on AWS with Terraform

Following is a step by step guide to installing CircleCI Server v2.17 with Terraform.

## Define Variables for Terraform

1. Clone the [Setup](#) repository. If you already have it cloned, make sure it is up-to-date and you are on the `master` branch by running:

```
git checkout master && git pull
```

2. Go to the top directory of the `enterprise-setup` repo on your local machine.
3. Run `terraform init` to initialize your working directory.
4. Run `make init` to initialize a `terraform.tfvars` file (your previous `terraform.tfvars` if any, will be backed up in the same directory).
5. Open `terraform.tfvars` in an editor and fill in appropriate AWS values for section 1.
6. If you plan to use 1.0 builders, specify a `circle_secret_passphrase` in section 2, replacing `...` with alpha numeric characters, if not, leave it as is. 1.0 builders are disabled by default in section 3.
7. Specify the instance type to use for your Nomad clients. By default, the value specified in the `terraform.tfvars` file for Nomad Clients is `m4.2xlarge` (8 vCPUs, 32GB RAM). To increase the number of concurrent CircleCI jobs that each Nomad Client can run, modify section 2 of the `terraform.tfvars` file to specify a larger `nomad_client_instance_type`. Refer to the AWS [Amazon EC2 Instance Types](#) guide for details.



The `builder_instance_type` is only used for CircleCI 1.0 and is disabled by default in section 3.

8. In section 3 you can:
  - a. choose to use 1.0 Builders if your project requires it (by changing the count to `1`)
  - b. enter proxy details, and enter a prefix if there will be multiple installations within your AWS region – the Services and Nomad client instances will be displayed with this prefix in the AWS console.

```
#####
# 1. Required Cloud Configuration
#####

aws_access_key = "..."
aws_secret_key = "..."
aws_region = "eu-central-1"
aws_vpc_id = "..."
aws_subnet_id = "..."
aws_ssh_key_name = "..."

#####
# 2. Required CircleCI Configuration
#####

circle_secret_passphrase = "..."
services_instance_type = "m4.2xlarge"
builder_instance_type = "r3.4xlarge"
nomad_client_instance_type = "m4.2xlarge"

#####
# 3. Optional Cloud Configuration
#####

# Set this to `1` or higher to enable CircleCI 1.0 builders
desired_builders_count = "0"

# Provide proxy address if your network configuration requires it
http_proxy = ""
https_proxy = ""
no_proxy = ""

# Use this var if you have multiple installation within one AWS region
prefix = "..."

services_disable_api_termination = "false"
force_destroy_s3_bucket = "true"
```

Figure 1. Example tfvars

Above is an example of the `terraform.tfvars` file you will be editing. The table below shows some of the default settings, and some optional variables that can be used to further customize your cluster. A full list of

variables and defaults can be found in the `variables.tf` file in the root of the `enterprise-setup` directory.

Optional vars:

Var	Description	Default
<code>services_instance_type</code>	Instance type for the centralized services box. We recommend a m4 instance	m4.2xlarge
<code>builder_instance_type</code>	Instance type for the 1.0 builder machines. We recommend a r3 instance	r3.2xlarge
<code>max_builders_count</code>	Max number of 1.0 builders	2
<code>nomad_client_instance_type</code>	Instance type for the nomad clients (2.0 builders). We recommend a XYZ instance	m4.xlarge
<code>max_clients_count</code>	Max number of nomad clients	2
<code>prefix</code>	Prefix for resource names	circleci
<code>enable_nomad</code>	Provisions a nomad cluster for CircleCi Server v2.x	1
<code>enable_route</code>	Enable creating a Route53 route for the Services box	0
<code>services_user_data_enabled</code>	Set to 0 to disable automated installation on Services Box	1
<code>force_destroy_s3_bucket</code>	Add/Remove ability to forcefully destroy S3 bucket when your installation is shut down	false
<code>services_disable_api_termination</code>	Protect the services instance from API termination. Set to false if you would like to terminate the Services box automatically when your installation is shut down	true

## Provision Instances

1. Save your changes to the `tfvars` file and run the following:

```
terraform plan
```

2. To provision your instances, run the following:

```
terraform apply
```

You will be asked to confirm if you wish to go ahead by typing **yes**.

3. An IP address will be provided at the end of the Terraform output. Visit this IP to carry on the install process.

## Access Your Installation

1. You will see a browser-specific SSL/TLS info box. This is just to inform you that on the next screen your browser might tell you the connection to the admin console is unsafe, but you can be confident it is secure. Click Continue to Setup and proceed to your installation IP.

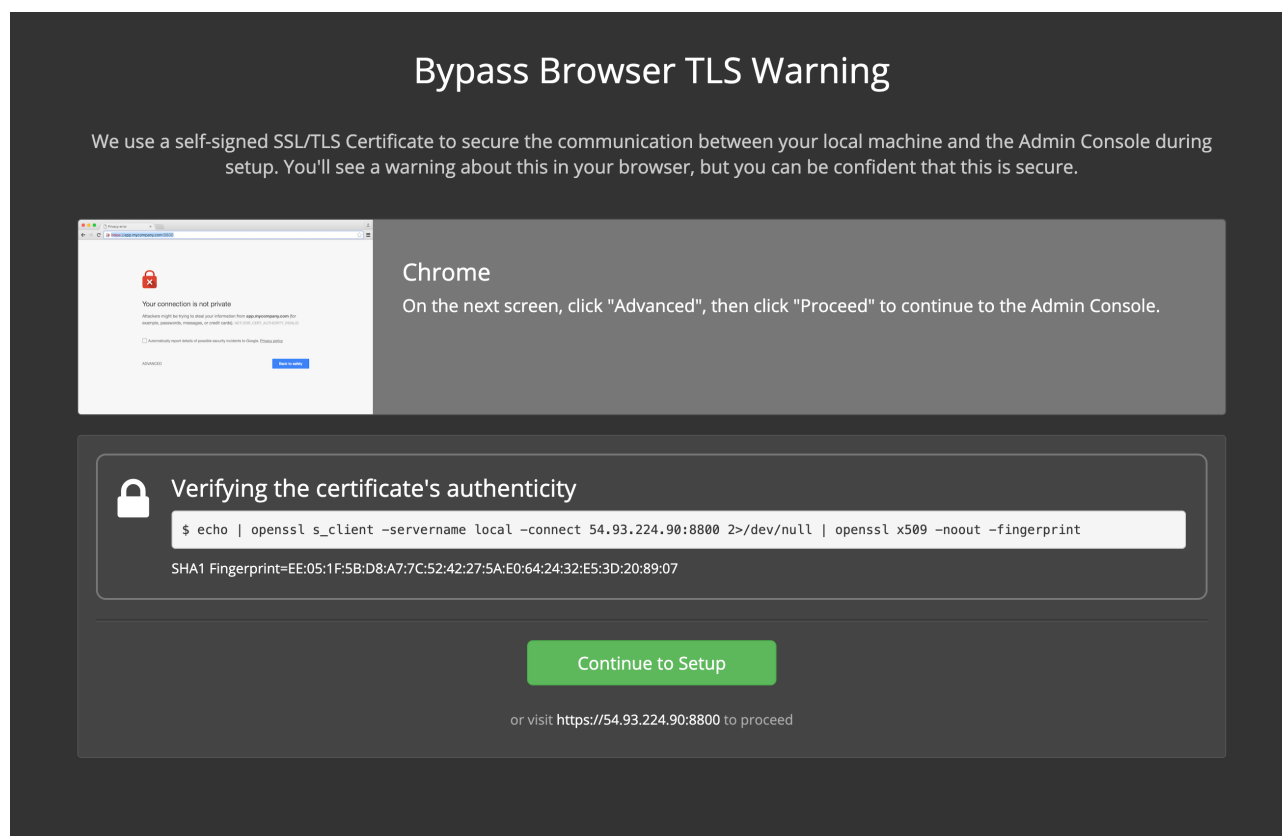


Figure 2. SSL Security

2. Enter your hostname – this can be your domain name or public IP of the Services Machine instance. At this time you can also upload your SSL public key and certificate if you have them. To proceed without providing these click Use Self-Signed Cert – choosing this option will mean you will see security warnings each time you visit the Management Console.



**HTTPS for admin console**

We're currently using a self-signed TLS certificate to secure the communication between your browser & the management console. If you don't upload your own TLS cert, you'll see a warning about this in your browser every time you access the management console.

**Provide Custom SSL Certificate**

Hostname (Ensure this domain name resolves to this server & is routable on your network)

app.yourdomain.com

Private Key

Choose file

Certificate

Choose file

Files will be uploaded directly to the management server & will never leave.  
If your private key and cert are already on this server, click here.

Use Self-Signed Cert    Upload & Continue

Figure 3. Hostname

3. Upload your license.
4. Decide how to secure the Management Console. You have three options:
  - a. Anonymous admin access to the console, anyone on port 8800 can access (not recommended)
  - b. Set a password that can be used to securely access the Management Console (recommended)
  - c. Use your existing directory-based authentication system (for example, LDAP)

## Secure the Admin Console

Keeping this admin console secure is important.  
You can create a shared password that will be required to access the settings, or you can connect it to your existing directory based authentication system.

☐ Anonymous ☒ Password ☐ LDAP

Password

Confirm Password

Continue

Figure 4. Admin Password

5. Your CircleCI installation will be put through a set of preflight checks, once they have completed, scroll down and click Continue.

## Preflight Checks

- ✓ **Successful HTTP request**  
Can access api.replicated.com
- ✓ **OS linux is supported**  
The operating system must be linux
- ✓ **Kernel version requirement met**  
Kernel version must be at least 3.10
- ✓ **Successful TLS connection**  
Can connect to TLS 172.31.23.155 address
- ✓ **Total space requirement met for directory /tmp**  
Directory must have at least 1G total space
- ✓ **Total space requirement met for directory /var/lib/replicated**  
Directory must have at least 250M total space
- ✓ **Docker server version requirement met**  
Docker server version must be exactly 17.12.1
- ✓ **CPU cores requirement met**  
Server must have at least 2 CPU cores
- ✓ **Memory requirement met**  
Server must have at least 8G total memory
- ✓ **Total space requirement met for directory /**  
Directory must have at least 60G total space
- ✓ **Total space requirement met for directory /var/lib/docker**  
Directory must have at least 1G total space
- ✓ **Successful Docker registry ping**  
Can access registry registry.replicated.com
- ✓ **Successful Docker registry ping**  
Can access registry index.docker.io

Node: b210b1a2dd95...

- ✓ **OS linux is supported**  
The operating system must be linux

Figure 5. Preflight Checks

# Installation Setup

You should now be on the Management Console settings page (your-circleci-hostname.com:8800).



You can make changes to the settings on this page at any time but changes here will require **downtime** while the service is restarted. Some settings are covered in more detail in our Operations Guide.

1. The Hostname field should be pre-populated from earlier in the install process, but if you skipped that step, enter your domain or public IP of the Services machine instance. You can check this has been entered correctly by clicking Test Hostname Resolution.
2. The Services section is only used when externalizing services. Externalization is available with a Platinum service contract. Contact [support@circleci.com](mailto:support@circleci.com) if you would like to find out more.

Figure 6. External Services

3. Under Execution Engines, only select 1.0 Builders if you require them for a legacy project – most users will leave this unchecked.
4. Select Cluster in the 2.0 Builders Configuration section. The Single box option will run jobs on the Services machine, rather than a dedicated instance, so is only suitable for trialling the system, or for some small teams.

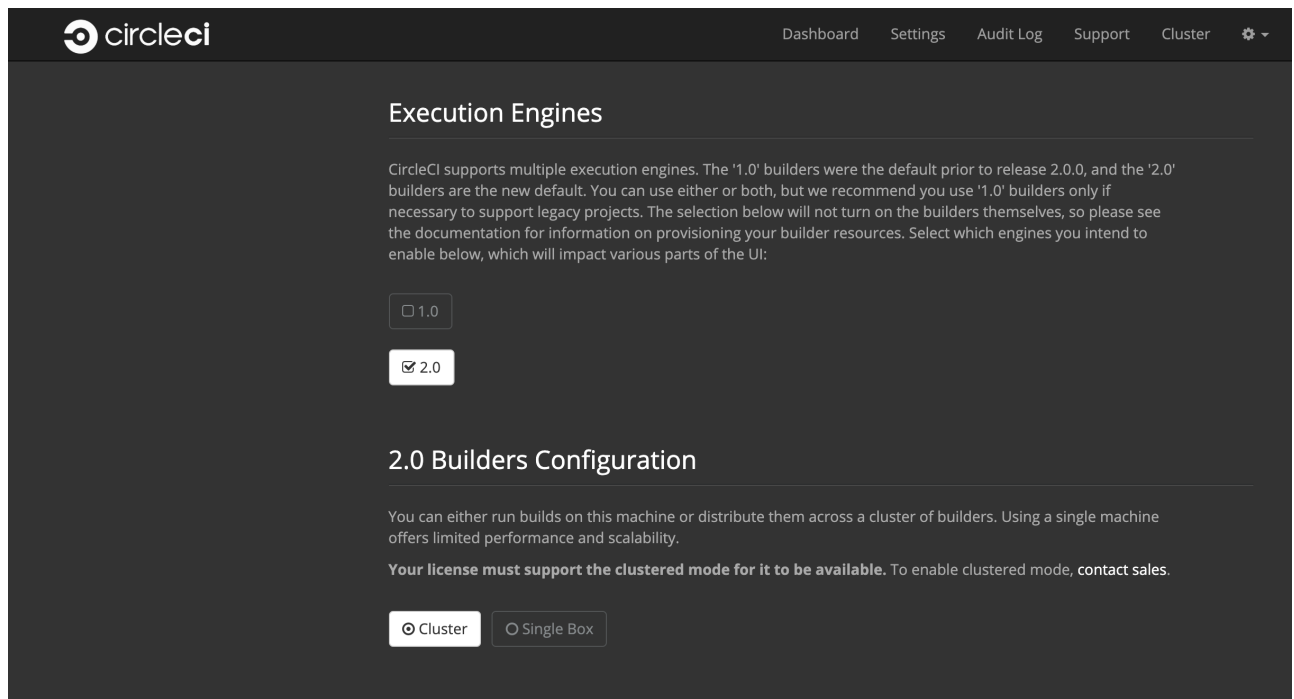


Figure 7. 1.0 and 2.0 Builders

5. Register CircleCI as a new OAuth application in GitHub.com or GitHub Enterprise by following the instructions provided onscreen.



If you get an "Unknown error authenticating via GitHub. Try again, or contact us." message, try using [http:](#) instead of [https:](#) for the Homepage URL and callback URL.

6. Copy the Client ID and Secret from GitHub and paste it into the relevant fields, then click Test Authentication.
7. If you are using GitHub.com, move on to the next step. If using Github Enterprise, you will also need to supply an API Token so we can verify your organization. To provide this, complete the following from your GitHub Enterprise dashboard:
  - a. Navigate to Personal Settings (top right) > Developer Settings > Personal Access Tokens.
  - b. Click "generate new token". Name the token appropriately to prevent accidental deletion. Do not tick any of the checkboxes, we only require the default public read-level access so no extra permissions are required. We recommend this token should be shared across your organization rather than being owned by a single user.
  - c. Copy the new token and paste it into the GitHub Enterprise Default API Token field.

**GitHub Integration**

CircleCI integrates with github.com or GitHub Enterprise. You will need to create an "OAuth Application" for CircleCI by doing the following:

1. Go to <https://github.com/settings/applications/new> for github.com or <https://<your GitHub hostname>/settings/applications/new> for GitHub Enterprise
2. Enter an application name like "CircleCI"
3. Enter "http(s)://<your hostname>" as the homepage URL
4. Enter "http(s)://<your hostname>/auth/github" as the authorization callback URL.

**NOTE:** The hostname and http(s) protocol on the callback URL must match the hostname and SSL settings on this page. **NOTE:** CircleCI does not support changing the URL or Backend Github instance after it has been set. Doing so can potentially take down your instance of CircleCI. If this is a hard requirement then please contact us. Not following the instructions could lead to catastrophic system failure.

☐ Public GitHub ☒ GitHub Enterprise

GitHub Enterprise Domain (Required)

The hostname associated with GitHub Enterprise, e.g. ghe.example.com

☒ HTTPS (TLS/SSL Enabled) ☐ HTTP (unencrypted/insecure) ☐ HTTPS (with self signed)

GitHub Application Client ID (Required)

GitHub Application Client Secret (Required)

GitHub Enterprise Default API Token (Required)

A GitHub token to use for requests when no OAuth token is available.

Figure 8. Enter Github Enterprise Token

8. If you wish to use LDAP authentication for your installation, enter the required details in the LDAP section.
9. We recommend using an SSL certificate and key for your install. You can submit these in the Privacy section if this step was missed during the installation.

**Privacy**

☒ SSL only (Recommended)

Forces TLS/SSL for your installation. A valid x509 SSL certificate and private key files are required to use this option. The certificate and key must be in PEM format. The key must be *unencrypted*.

Certificate File (Required)

Private Key File (Required)

Figure 9. Privacy Settings

10. We recommend using S3 for storage and all required fields for Storage are pre-populated. The IAM user, as referred to in the [planning](#) section of this document, is used here.

**Storage**

CircleCI supports multiple cloud providers. We recommend using the native object storage of your provider to store build artifacts and files, which requires your cloud credentials for authenticating with the APIs. You can choose to use our experimental local object storage, but we do not recommend this for production systems.

☒ AWS S3 ☐ None

CircleCI will use an S3 bucket for storing build-related artifacts. Supply the AWS keys here.

To create an IAM user/role with the proper permissions, you can follow the [Getting Started - AWS document](#). (The provided Terraform or CloudFormation resources can automatically create an IAM role and instance profile.)

**AWS Region** (Required)

eu-central-1

**S3 Bucket** (Required)

rosie-bucket-44b34e2c

The bucket will be created if necessary.

☒ IAM Instance Profile ☐ IAM User (Key+Secret)

**AWS Authentication**

Figure 10. Storage Options

11. Complete enhanced AWS Integration options.
12. Complete the Email section if you wish to configure your own email server for sending build update emails. Leave this section is you wish to use our default email server.



Due to an issue with our third party tooling, Replicated, the Test SMTP Authentication button is not currently working

13. Configure VM service if you plan to use [Remote Docker](#) or [machine](#) executor features. We recommend using an IAM instance profile for authentication, as described in the [planning](#) section of this document. With this section completed, instances will automatically be provisioned to execute jobs in Remote Docker or use the [machine](#) executor. For more information on VM Service and creating custom AMIs for remote Docker and [machine](#) executor jobs, see our [VM service guide](#).

**circleci** Dashboard Settings Audit Log Support Cluster

## VM Provider

Configure automated provisioning of Virtual Machines to enable users to use Remote Docker or the **machine** executor.

☐ None
 ☒ **AWS EC2**
☐ On-Host

We use your EC2 credentials to automatically provision boxes on demand when users request Remote Docker or the **machine** executor.

**AWS Region (Required)**  
eu-central-1

**EC2 Subnet ID (Required)**  
subnet-5fc60125  
AWS EC2 Subnet ID to be used for VM creation. Should be in the region specified above.

**EC2 Security Group ID (Required)**  
sg-083751620e2453e24  
AWS EC2 Security Group ID to be assigned for all VMs. Should be in the region specified above.

**Custom VM AMI**  
  
AMI to be used for machine executor and remote docker VMs instead of default. Should be in the selected region and available for AWS User specified above.

**AWS Instance Type (Required)**  
t2.medium  
Instance type the VM services should use.

**AWS Authentication**  
☒ IAM Instance Profile
 ☐ IAM User (Key+Secret)

**VM Preallocation**  
Number of VMs to preallocate and have waiting to execute jobs. Set to 0 to have only on-demand VM provisioning.

**Remote Docker**  
0

**Machine Executor**  
0

☐ Show Advanced Settings

Figure 11. Configure VM Service

You can preallocate instances to always be up and running, reducing the time taken for Remote Docker and **machine** executor jobs to start. If preallocation is set, a cron job will cycle through your preallocated instances once per day to prevent them getting into a bad/dead state.



If Docker Layer Caching (DLC) is to be used VM preallocation must be set to **0** – on-demand – for both Remote Docker and **machine** executor.

- If you wish to use AWS Cloudwatch or Datadog for collating metrics for your installation, set this up here. For more information see our [Monitoring guidance](#):

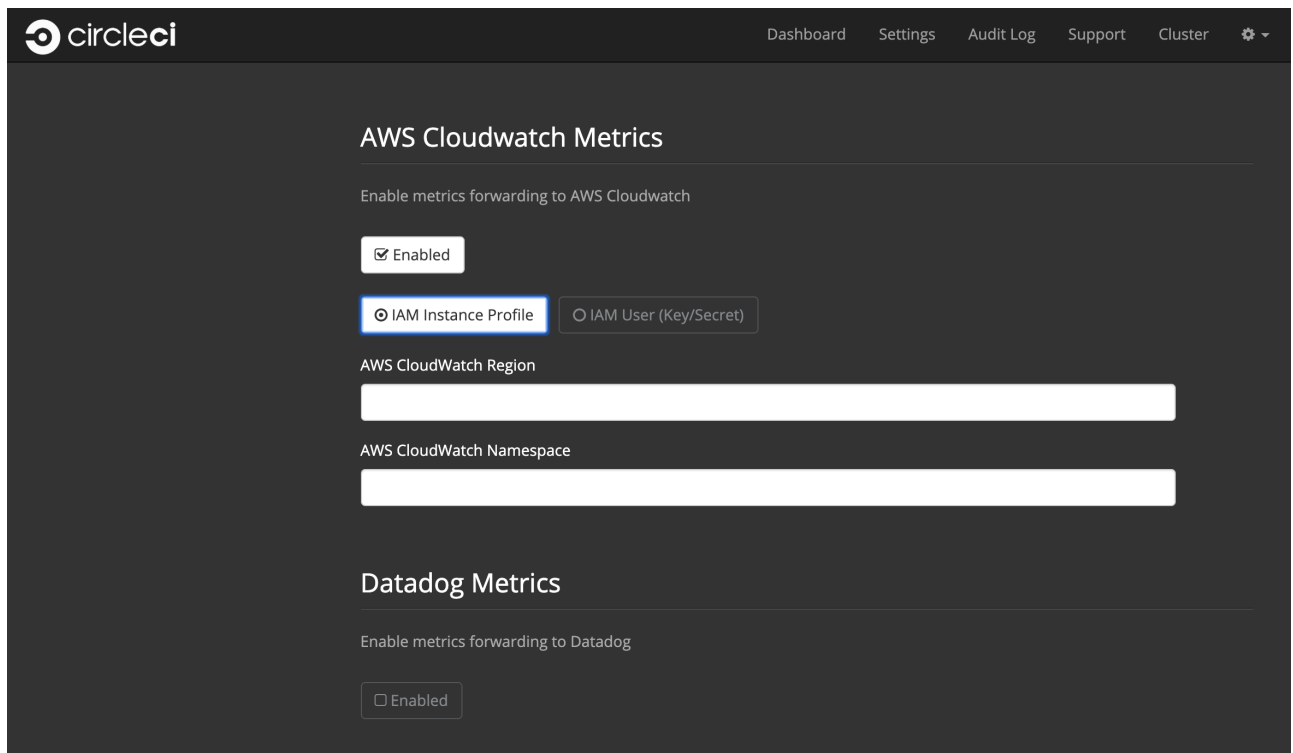
The screenshot shows the CircleCI management console with a dark theme. At the top is a navigation bar with the CircleCI logo and links for Dashboard, Settings, Audit Log, Support, and Cluster. The main content area is titled 'AWS Cloudwatch Metrics' and includes a toggle for 'Enable metrics forwarding to AWS Cloudwatch' which is currently 'Enabled'. Below this are two radio button options: 'IAM Instance Profile' (which is selected and highlighted with a blue border) and 'IAM User (Key/Secret)'. There are also two text input fields for 'AWS CloudWatch Region' and 'AWS CloudWatch Namespace'. A second section titled 'Datadog Metrics' has a toggle for 'Enable metrics forwarding to Datadog' which is currently 'Disabled'.

Figure 12. Metrics

You can also customize the metrics received through Telegraf. For more on this see our [Custom Metrics](#) guide.

15. Artifacts persist data after a job is completed, and may be used for longer-term storage of your build process outputs. By default, CircleCI Server only allows approved types to be served. This is to protect users from uploading, and potentially executing malicious content. The **Artifacts** setting allows you to override this protection. For more information on safe/unsafe types see our [Build Artifacts guidance](#).
16. After agreeing to the License Agreement and saving your settings, select Restart Now from the popup. You will then be redirected to start CircleCI and view the Management Console Dashboard. It will take a few minutes to download all of the necessary Docker containers.



If the Management Console reports `Failure reported from operator: no such image` click Start again and it should continue.

## Validate Your Installation

1. When the application is started, select Open to launch CircleCI in your browser, and sign up/log in to your CircleCI installation and start running 2.0 builds! You will become the Administrator at this point as you are the first person to sign in. Have a look at our [Getting Started](#) guide to start adding projects.



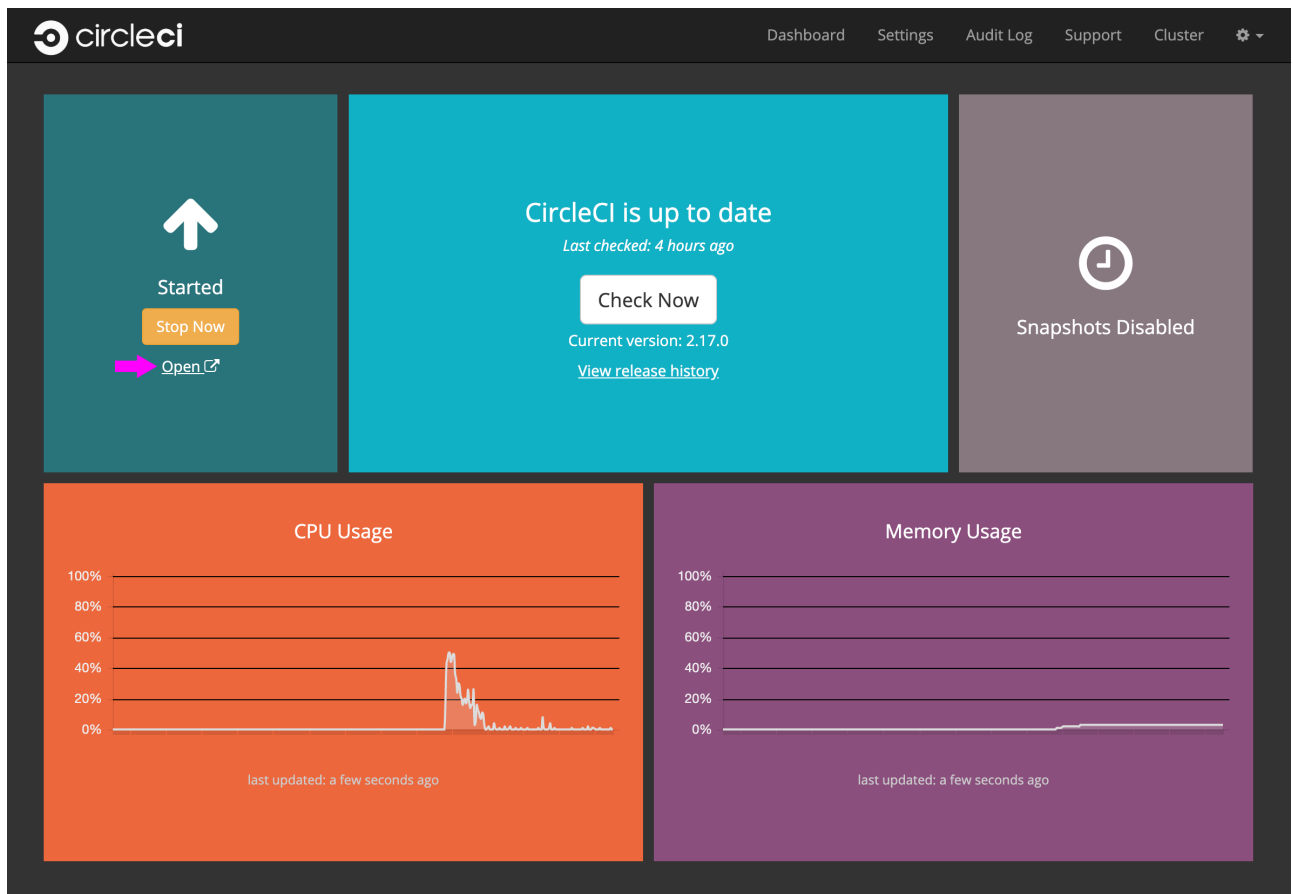


Figure 13. Start CircleCI from your Dashboard

2. After build containers have started and images have been downloaded, the first build should begin immediately. If there are no updates after around **15 minutes**, and you have clicked the Refresh button, contact [CircleCI support](#) for assistance.
3. Next, use [our realitycheck repo](#) to check basic CircleCI functionality.
4. If you're unable to run your first builds successfully please start with our [Troubleshooting](#) guide for general troubleshooting topics, and our [Introduction to Nomad Cluster Operation](#) for information about how to check the status of Builders in your installation.

# Teardown

If you wish to delete your installation of CircleCI Server, please let us know first in case there are any specific, supplementary steps required for your installation. Below is our basic step by step guide to tearing down an installation of CircleCI Server that was made with Terraform:

1. First you need to manually disable the termination protection on the Services machine from the AWS Management Console (If you set `services_disable_api_termination = "false"` in your `terraform.tfvars` file, skip this step). To do this:
  - a. Navigate to the EC2 Dashboard and locate the Services machine instance
  - b. Click to select it
2. Click Actions > Instance Settings > Change Termination Protection
3. Navigate to the S3 dashboard, locate the S3 bucket associated with your CircleCI cluster and delete the bucket and its contents (If you set `force_destroy_s3_bucket = "true"` in your `terraform.tfvars` file, skip this step).
4. From a terminal, navigate to your clone of our `enterprise-setup` repo and run `terraform destroy` to destroy all EC2 instances, IAM roles, ASGs and Launch configurations created by `terraform apply`.

# Upgrading a Server Installation

This section describes the process for upgrading your CircleCI Server installation from v2.17 to v2.18.

## Org Rename Script



Before upgrading please read and follow the steps below if you have **ever had issues with renaming an organization within CircleCI** or you suspect that an **organization rename** might have happened at any point.

1. SSH into your Services machine
2. REPL into `workflows-conductor` by running the following: `sudo docker exec -it workflows-conductor lein repl :connect 6005`
3. Copy/paste this script into the REPL session. It will run migration and output current progress.

```
(def domain-client @workflows-conductor.domain.contexts/*domain-connection*)
(def permissions-client @workflows-conductor.domain.contexts/*permissions-connection*)

;; Syncing orgs
(require '[circleci.domain.client.orgs :as domain-orgs])

(defn get-domain-org
  "Pass-through fn that actually makes API call to domain-service"
  [org-id]
  (domain-orgs/org-by-id domain-client org-id))

(defn new-domain-org
  "Pass-through fn that actually makes API call to domain-service"
  [org-data]
  (domain-orgs/new-org domain-client org-data))

(defn sync-org
  [org]
  (when org
    (let [domain-org (try
                      (get-domain-org (:analytics-id org))
                      (catch Exception e
                        (println "ERROR: Exception while fetching
org from domain-service" (:analytics-id org))
                        (throw e))))])
```

```

        mongo-ids {:id (:analytics-id org)
                   :external-id (str (or (:github-id org)
                                         (:bitbucket-id org)))}
        domain-ids (select-keys domain-org [:id :external-id])
        (when-not (= mongo-ids domain-ids)
          (let [new-domain-org (try
                                (new-domain-org {:id (:analytics-id
                                                         org)
                                                  :name (:name org)
                                                  :provider-id (:id
                                                                (circle.services.domain/provider-by-type (:vcs-type org)))
                                                  :external-id (some->
                                                                (or (:github-id org) (:bitbucket-id org))
                                                                str))
                                (catch Exception e
                                  (println "ERROR: Exception while
creating an org" (:analytics-id org))
                                  (throw e)))
              new-domain-ids (select-keys new-domain-org [:id
                                                           :external-id])]
            (when-not (= mongo-ids new-domain-ids)
              (println "ERROR: Orgs doesn't match after sync.\nMongo
org:" mongo-ids "\ndomain-service org:" domain-ids))))))

(defn sync-all-orgs
  [last-id]
  (loop [prev-id last-id]
    (let [org (first (circle.model.organization/find-active :where
                                                            (when prev-id {:_id {:$gt prev-id}})
                                                            :only
                                                            [:github-id :bitbucket-id :analytics-id :name :vcs-type]
                                                            :limit 1
                                                            :sort
                                                            {:_id 1})))]
      (when org
        (println "Syncing org" (select-keys org [:_id :name]))
        (try
          (sync-org org)
          (catch Exception e
            (println e))))))

```

```

(recur (:_id org))))))

;; Syncing contexts
(require '[circleci.permissions.client.actors :as actors])
(require '[circleci.permissions.client.permissions :as permissions])
(require '[circleci.permissions.shared.permissions :as permissions-
names])

(defn context-to-permission [{:contexts-service-
client.context.response/keys [id name organization-ref] :as context}]
  (println (format "\tCreating permissions for context '%s' (%s)"
name id))
  (try
    (if-not (circleci.domain.client.contexts/context-by-id domain-
client id)
      (println (format "ERROR: context '%s' (%s) doesn't exist in
domain-service") name id)
      (when (empty? (actors/actors-allowed-to-run-contexts
permissions-client [id]))
        (permissions/new-permission permissions-client organization-
ref permissions-names/run-context id)
        (permissions/new-permission permissions-client organization-
ref permissions-names/execute-context id)
        (permissions/new-permission permissions-client organization-
ref permissions-names/view-context id)
        (permissions/new-permission permissions-client organization-
ref permissions-names/edit-context-vars id)))

    (catch Exception e
      (println (format "\tERROR: Unexpected error while creating
permissions: '%s'" (.getMessage e))))))

(defn context-to-domain
  [{:contexts-service-client.context.response/keys [id name
organization-ref] :as context}]
  (try
    (println (format "\tMigrating context '%s' (%s)" name id))
    (circleci.domain.client.contexts/new-context domain-client id
name organization-ref nil)

    (catch clojure.lang.ExceptionInfo e

```

```

    (let [{:keys [status body]} (ex-data e)
          {:keys [errors]} (cheshire.core/parse-string body true)]
      (cond
        ;; context already migrated
        (and (= status 400)
              (= 1 (count errors))
              (= "key_duplication" (some-> errors first :type)))
        nil

        ;; org not found
        (and (= status 400)
              (= 1 (count errors))
              (= "bad_argument" (some-> errors first :type))
              (= "org_id" (some-> errors first :value)))
        (println "\tERROR: domain-service doesn't know org with id"
                  organization-ref)

        :else
        (println "\tERROR: Failed to create a context in domain-
service:" (ex-data e))))))

    (catch Exception e
      (println (format "\tERROR: Unexpected error while creating
context in domain-service: '%s'" (.getMessage e))))))

(defn migrate-contexts
  []
  (doseq [org (circle.model.organization/find-active :where {} :only
[:analytics-id :name])]
    (println (format "Migrating contexts for org '%s' (%s)" (:name
org) (:analytics-id org)))
    (doseq [context (contexts-service-client.core/list-contexts
@workflows-conductor.domain.contexts/*contexts-connection*
(:analytics-id org))]
      (context-to-domain context)
      (context-to-permission context))))

(defn migrate
  []
  (println "==== Syncing orgs =====\n")
  (sync-all-orgs nil))

```

```
(println "\n=== Syncing contexts ===\n")
(migrate-contexts))

(migrate)
```

4. If any **ERROR** messages are present in the output please report back to your CSM or reach out to support.

## Upgrade Steps Overview

Following is an overview of the CircleCI Server upgrade steps. Each stage is described in detail below.

- Take a snapshot of your installation so you can rollback later if necessary (optional but recommended)
- Update Replicated and check you are running Docker v17.12.1, update if necessary
- Install the latest version of CircleCI Server

### 1. Snapshot for Rollback

To take a snapshot of your installation:

1. Go to the Management Console (e.g. [your-circleci-hostname.com:8800](#)) and click Stop Now to stop the CircleCI service.

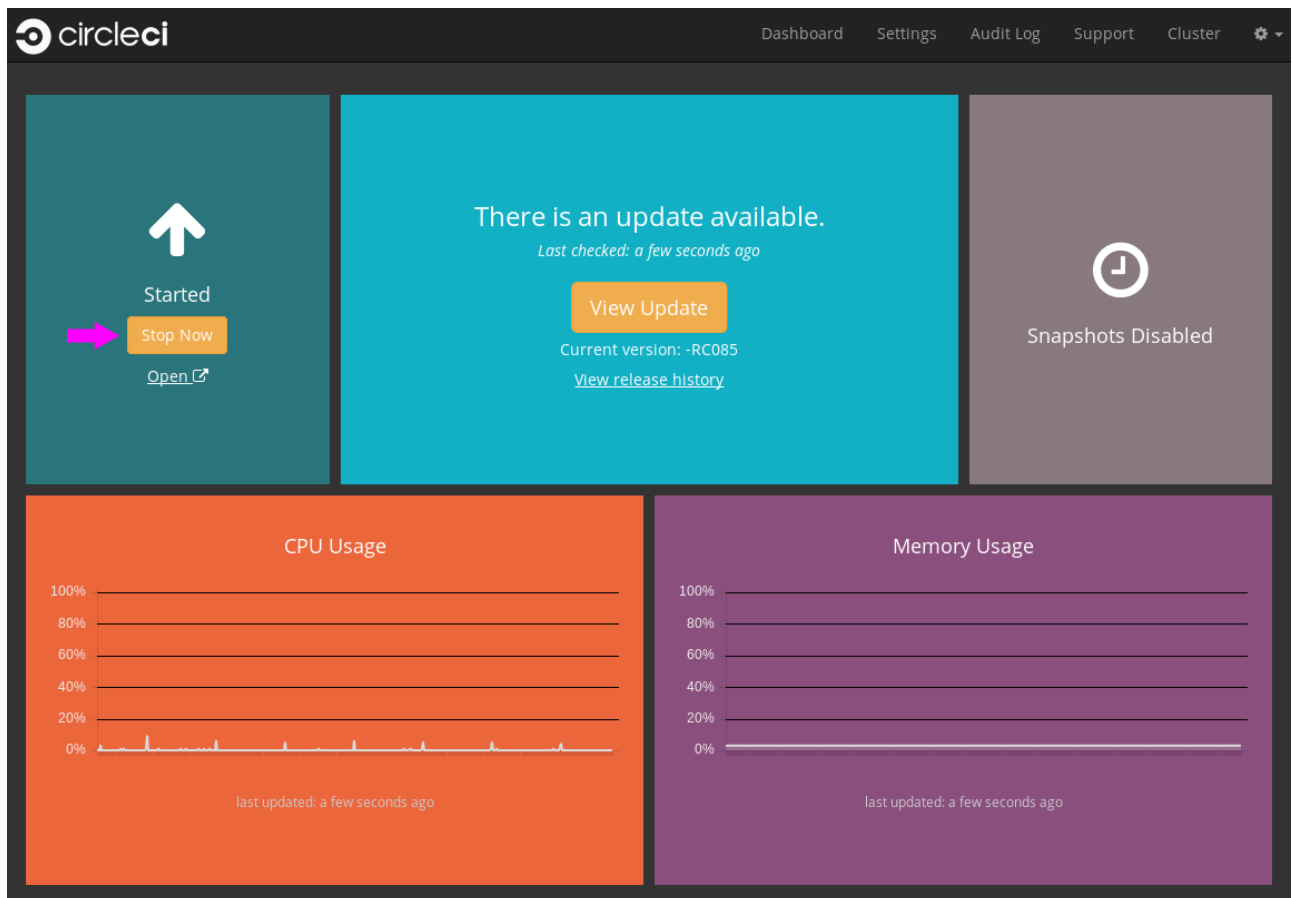


Figure 14. Stop CircleCI

2. Ensure no jobs are running on the nomad clients – you can check this by running `nomad status`
3. Navigate to the AWS EC2 management console and select your Services machine instance
4. Select Actions > Image > Create Image – Select the No Reboot option if you want to avoid downtime at this point. This image creation step creates an AMI that can be readily launched as a new EC2 instance to restore your installation.

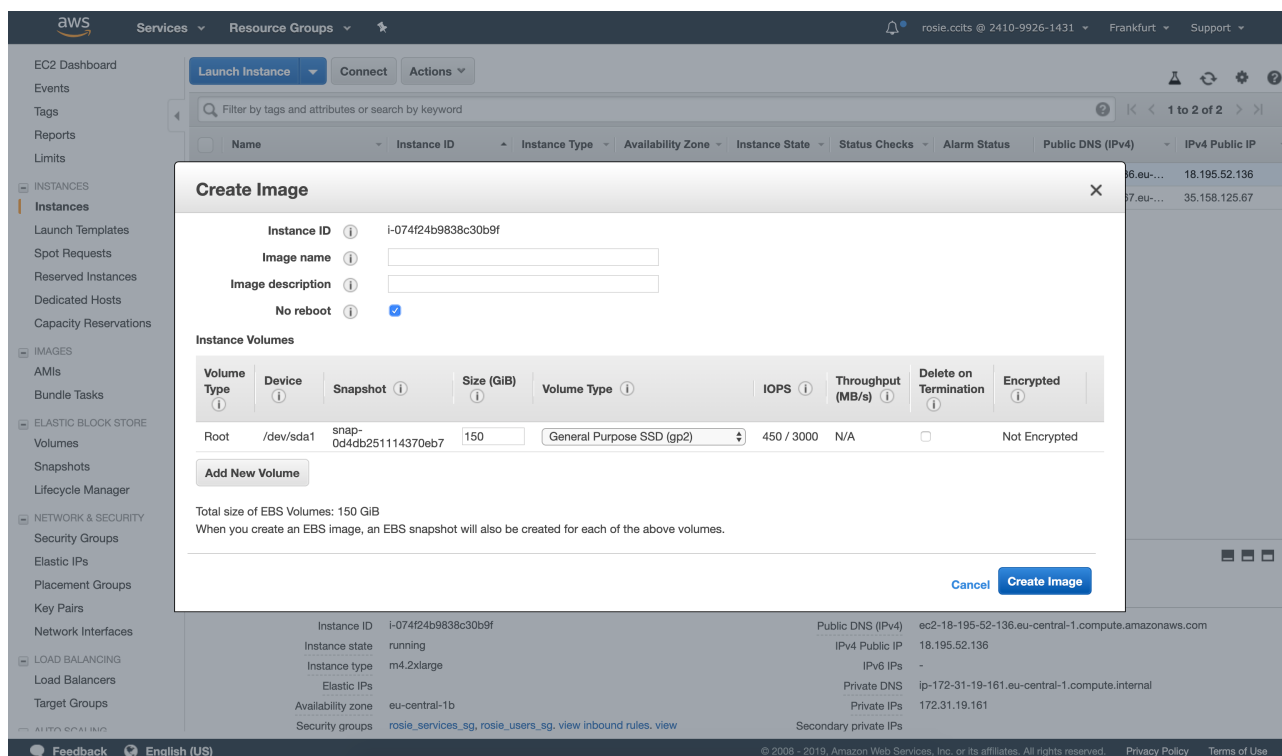


Figure 15. Snapshot Image Creation



It is also possible to automate this process with the AWS API. Subsequent AMIs/snapshots are only as large as the difference (changed blocks) since the last snapshot, such that storage costs are not necessarily larger for more frequent snapshots, see Amazon’s EBS snapshot billing document for details. Once you have the snapshot you are free to make changes on the Services machine.

If you do need to rollback at any point, see our [guide to restoring from a backup](#).

## 2. Updating Replicated

### Prerequisites

- Your installation is Ubuntu 14.04 or 16.04 based.
- Your installation is **not** airgapped and you can access the internet from it.
- You are running replicated version  $\geq 2.10.3$  on your services machine. To check this, SSH into the Services machine and run the following:

```
replicated --version
```



If you are running a version of Replicated pre 2.10.3 please reach out to [support@circleci.com](mailto:support@circleci.com).

## Preparations



Remember to take a snapshot (described above) before starting the Replicated update process

1. Stop the CircleCI application by clicking the Stop Now button on the Dashboard. Application shutdown takes a few minutes. Wait for the status to become “Stopped” before continuing.

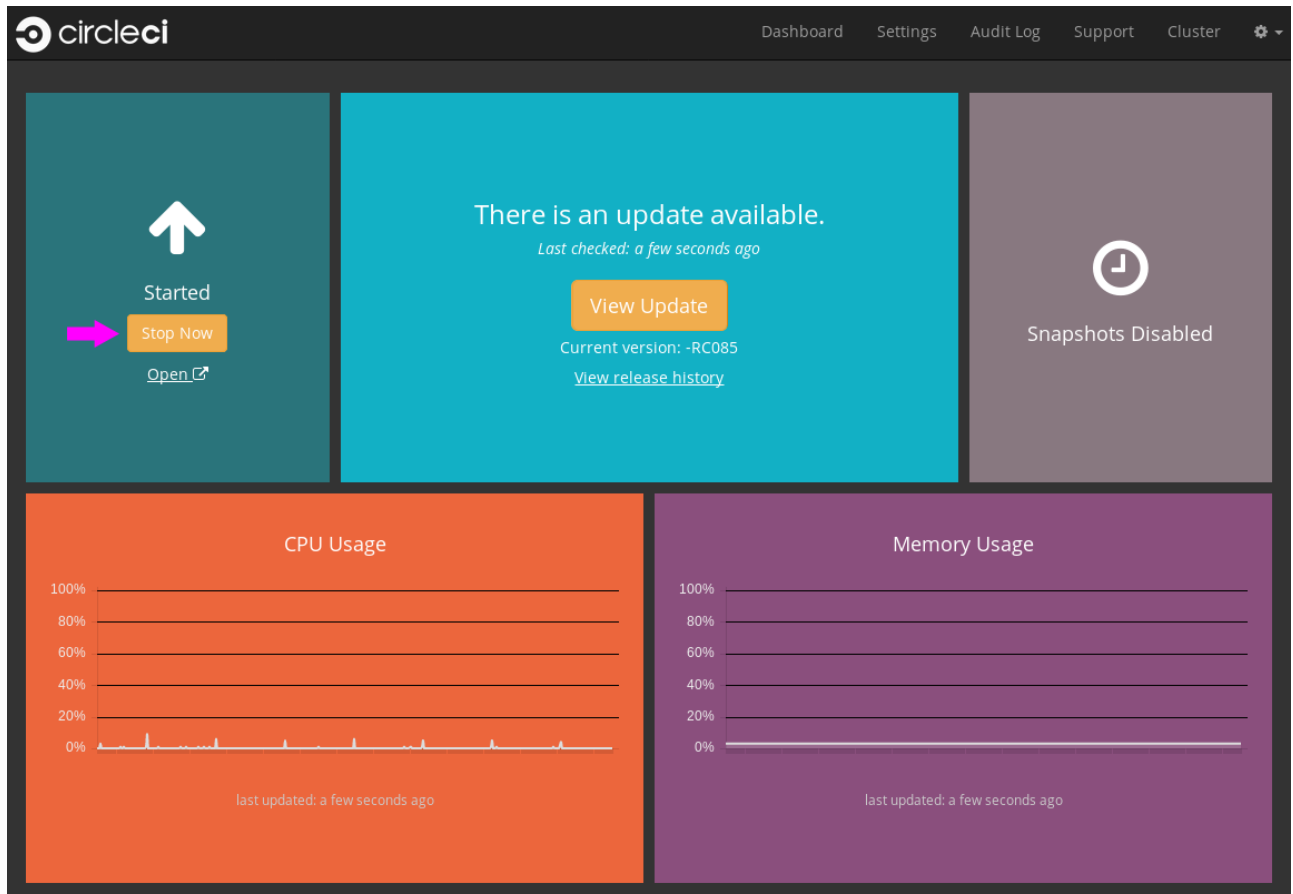


Figure 16. Stop the CircleCI Application

Alternatively you can SSH into the services machine and stop the CircleCI application from the command line:

```
replicatedctl app stop
```

You can check the status using the following:

```
replicatedctl app status inspect
```

Example Output:

```
[
  {
    "AppID": "edd9471be0bc4ea04dfca94718ddf621",
    "Sequence": 2439,
    "State": "stopped",
    "DesiredState": "stopped",
    "Error": "",
    "IsCancellable": false,
    "IsTransitioning": false,
    "LastModifiedAt": "2018-10-23T22:00:21.314987894Z"
  }
]
```

- For the replicated update to succeed, it is necessary to update docker to the recommended version, 17.12.1. Check which version you are running with `docker version` and if you need to update, follow these steps:

```
sudo apt-get install docker-ce=17.12.1~ce-0~ubuntu
```

- Pin the Docker version using the following command:

```
sudo apt-mark hold docker-ce
```

## Perform Update

- Perform the Replicated update by executing the update script as follows:

```
curl -sSL "https://get.replicated.com/docker?replicated_tag=2.38.0" |
sudo bash
```

Double-check your replicated and docker versions:

```
replicatedctl version    # 2.38.0
docker -v                # 17.12.1
```

- Restart the app with

```
replicatedctl app start
```

The application will take a few minutes to spin up. You can check the progress in the administration dashboard or by executing;

```
replicatedctl app status inspect
```

Example output:

```
[
  {
    "AppID": "edd9471be0bc4ea04dfca94718ddf621",
    "Sequence": 2439,
    "State": "started",
    "DesiredState": "started",
    "Error": "",
    "IsCancellable": true,
    "IsTransitioning": true,
    "LastModifiedAt": "2018-10-23T22:04:05.00374451Z"
  }
]
```

### 3. Upgrade CircleCI Server Version

1. Once you are running the latest version of Replicated, click the View Update button in the Management Console dashboard.

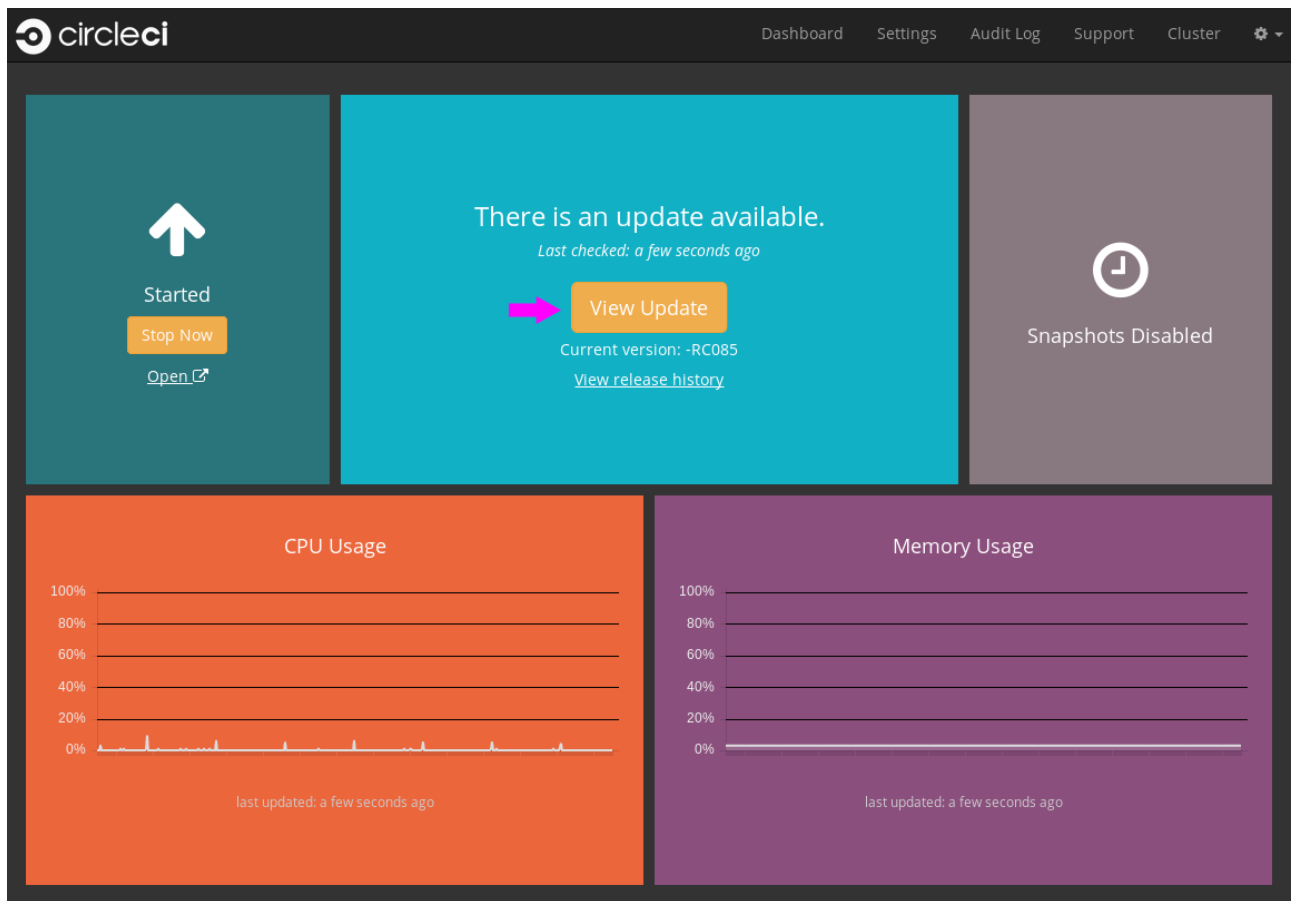


Figure 17. View Available Updates

2. Click Install next to the version you wish to install.

The install process may take several minutes and the install status will be displayed both on the Releases page and the main Dashboard.



Please refresh your screen intermittently during the install process to avoid unnecessary waiting.

3. Once the installation is finished, navigate to the Dashboard to start your installation - Note the middle box on the Dashboard will read "CircleCI is up to date" when you are running the latest version.