

INSTALLATION GUIDE

A guide for installing and upgrading CircleCI Server on AWS.

Docs Team

Version 2.17.2, 07-24-19

Installation Overview	1
Support Packages	2
Non-AWS Platform Support	2
Externalization	2
System Requirements	3
Services Machine	4
Nomad Clients	5
Server Ports	6
Services Machine	6
Nomad Clients	7
GitHub Enterprise / GitHub.com	8
PostgreSQL Servers	9
MongoDB Servers	9
RabbitMQ Servers	9
Redis Servers	10
Nomad Servers	10
Installation Prerequisites	11
Private Subnet Requirements	12
Planning	13
Installation on AWS with Terraform	15
Define Variables for Terraform	16
Provision Instances	19
Access Your Installation	19
Setup Your Installation	21
Validate Your Installation	24
Teardown	25

Installation Overview



CircleCI Server v2.17 uses the CircleCI 2.0 architecture.

This document provides step-by-step instructions for installing CircleCI Server v2.17 on Amazon Web Services (AWS) with Terraform. Refer to the [changelog](#) for what's new and fixed in this release.

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 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	

Port number	Protocol	Direction	Source / destination	Use	Notes
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
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	

Port number	Protocol	Direction	Source / destination	Use	Notes
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

We use 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 (`.rli`). 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, head [here](#) for a list of ports that need whitelisting.
- In order to provision required AWS entities with Terraform you need an IAM User with following permissions (for guidance on creating IAM users in your AWS account, head [here](#)):

```
{
  "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.

```
#####
# 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

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](<https://aws.amazon.com/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:

1. choose to use 1.0 Builders if your project requires it (by changing the count to `1`)
2. 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.

Figure 3 shows 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

Var	Description	Default
services_disable_api_termination	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 and run:

```
terraform plan
```

2. Next, to provision your instances, run:

```
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.

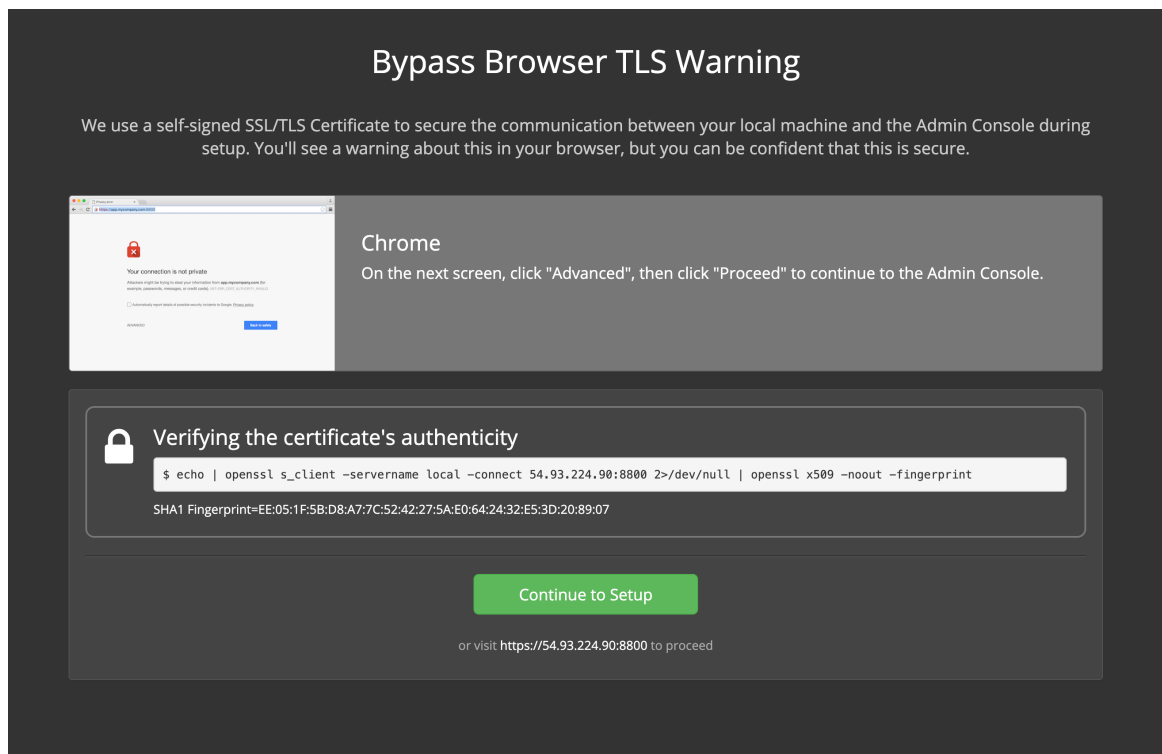


Figure 2. SSL Security

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.

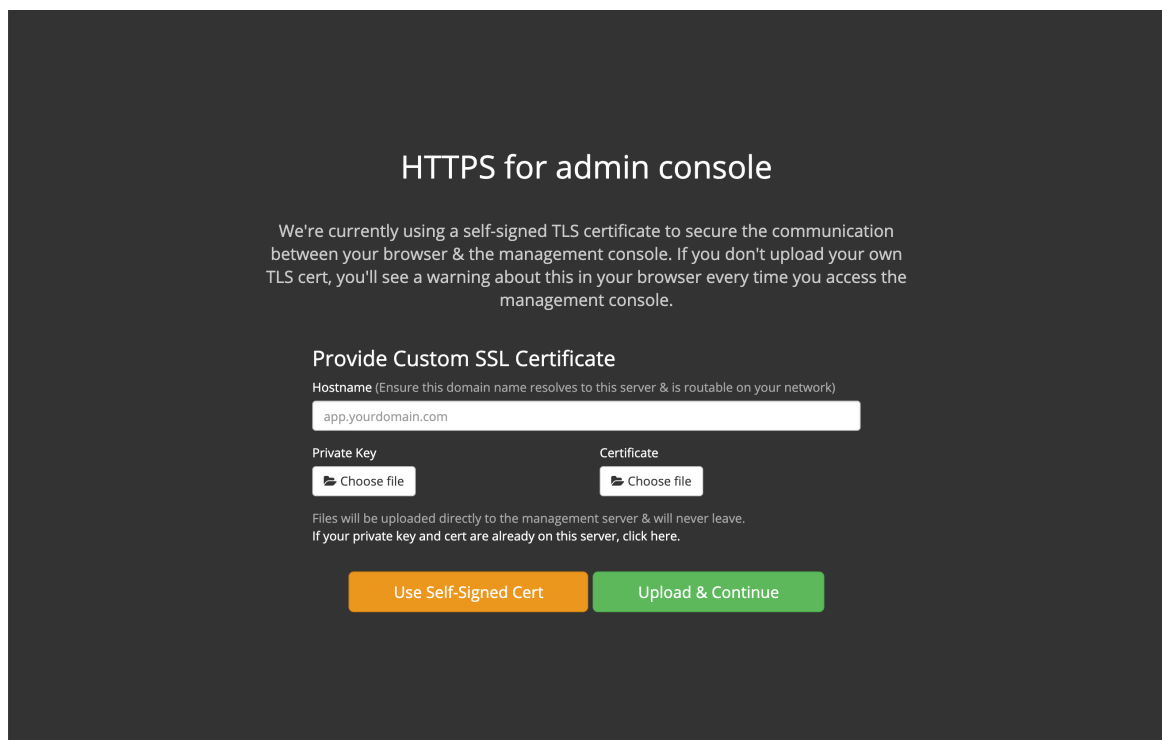
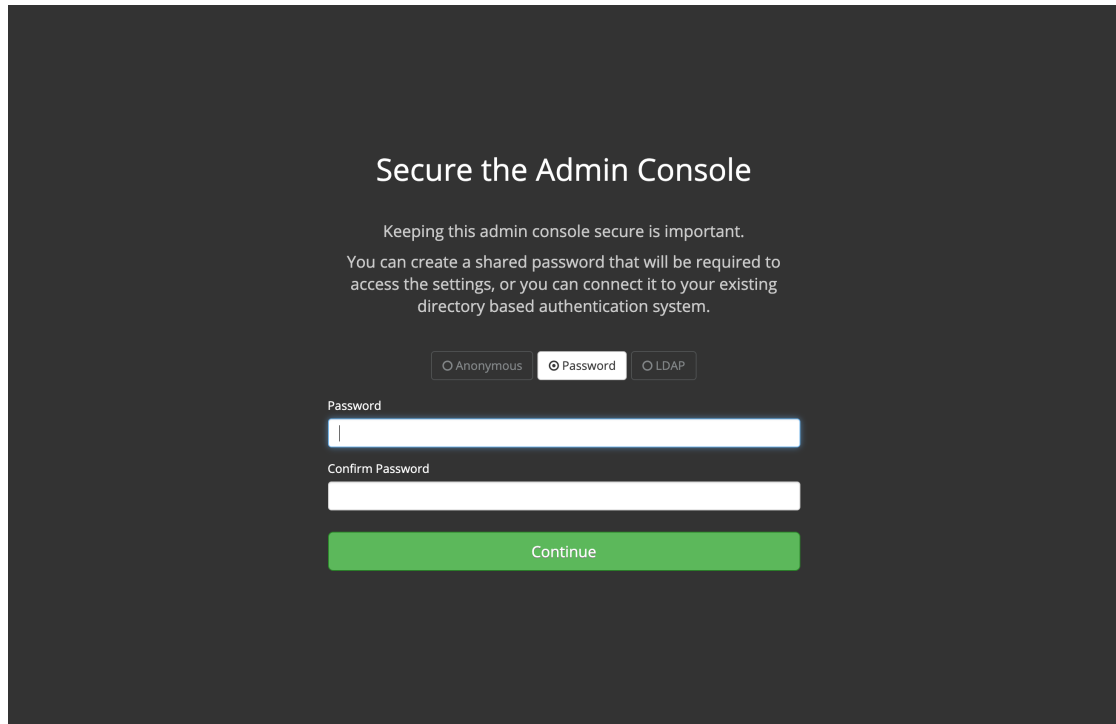


Figure 3. Hostname

3. Upload your license.

4. Decide how to secure the Management Console. You have three options:

- ☐ Anonymous admin access to the console, anyone on port 8800 can access (not recommended)
- ☐ Set a password that can be used to securely access the Management Console (recommended)
- ☐ Use your existing directory-based authentication system (e.g. LDAP)



5. Your CircleCI installation will be put through a set of preflight checks, once they have completed, scroll down and click Continue. (See figure 7) <!--what should admins do if not all these checks pass-->

```
![[Preflight Checks](images/preflight.png){ width=400px }
```

Setup Your Installation

You should now be on the Management Console settings page (<http://<your-circleci-hostname>.com:8800>). You can come back 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 if you would like to find out more.

```
! [Hostname and Services Settings](images/hostname-services.png){
width=400px }
```

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.

```
! [Execution Engine](images/builders.png){ width=400px }
```

5. Register CircleCI as a new OAuth application in GitHub.com or GitHub Enterprise by following the instructions provided onscreen. (See figure 10).

 **Note:** 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:
 1. Navigate to Personal Settings (top right) > Developer Settings > Personal Access Tokens.
 2. 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.
 3. Copy the new token and paste it into the GitHub Enterprise Default API Token field.

```
! [Github Integration](images/ghe_token.png){ width=400px }
```

8. LDAP TBD <!--insert LDAP instructions once I have more detail from Anton-->
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](images/privacy.png){ width=400px }
```

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


```
! [Storage](images/storage.png){ width=400px }
```

11. Enhanced AWS Integration TBD <!--explain enhanced AWS integration 1.0 or just say ignore-->
12. Email TBD <!--explain email server options, test button doesnt work etc-->
13. Configure the VM service – VM Provider section – if you plan to use [Remote Docker](<https://circleci.com/docs/2.0/building-docker-images/>) or **machine** executor features. We recommend using an IAM instance profile for authentication, as described in the \hyperref[sec:planning]{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.

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. ****Note****: 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. <!--Advanced settings to cover here too - make a section in ops guide for this and list of OK instance types for VM provider-->

```
! [VM Provider](images/vmprovider.png){ width=400px }
```

14. If you wish to use AWS Cloudwatch or Datadog for collating metrics for your installation, set this up here:

```
! [Metrics](images/metrics_setup.png){ width=400px }
```

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 whitelisted artifact types to be served. This is to protect users from uploading, and potentially executing malicious content. The **Artifacts** setting here allows you to override this protection. For more information on safe/unsafe types see the Build Artifacts chapter in the [CircleCI Server Operations Guide](<https://circleci.com/docs/2.0/circleci-ops-guide-v2-17.pdf#section=administration>)
16. After agreeing to the License Agreement and saving your settings, select Restart Now from the popup to get redirected to start the service 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

- 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](<https://circleci.com/docs/2.0/getting-started/#section=getting-started>) guide to start adding projects. <!--add info on making users administrators etc. to user management section of ops guide and put a link here-->

```
! [Dashboard](images/dashboard.png){ width=400px }
```

- 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](<https://support.circleci.com/hc/en-us>) for assistance.
- You can use [our realitycheck repo](<https://github.com/circleci/realitycheck>) to check basic CircleCI functionality. <!--Is this what we hope they will do? Could use stronger language. Also would be good to have some help for what to do if jobs fail, or do we just want people to tell us so we can help?-->
- If you're unable to run your first builds successfully please start with our [Operations Guide](<https://circleci.com/docs/2.0/circleci-ops-guide-v2-17.pdf#section=administration>), specifically the Troubleshooting section for general troubleshooting topics, and the Introduction to Nomad Cluster Operation for information about how to check the status of Builders in your installation.

\newpage

Teardown

If you wish to teardown your installation of CircleCI Server, please let us know [how?](#) 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:
 1. Navigate to the EC2 Dashboard and locate the Services machine instance
 2. Click to select it
 3. Click Actions > Instance Settings > Change Termination Protection
2. Navigate to the S3 dashboard, locate the S3 bucket associated with your CircleCI cluster and delete it/its contents (If you set `force_destroy_s3_bucket = "true"` in your `terraform.tfvars` file, skip this step).
3. 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`.