INSTALLATION GUIDE

A guide for installing or migrating to CircleCI Server v3.0.1 on AWS or GCP

docs@circleci.com

Version 3.0.1, 05/20/2021: FINAL

CircleCI Server v3.x Prerequisites

It is assumed you have already read the Server 3.x overview.

In order to configure the CircleCI Server application, you will need to ensure the following general and infrastructure-specific requirements are met. You will need:

- An existing Kubernetes cluster (see our guide if you need help creating one), for example:
  - Creating an Amazon EKS cluster - Amazon EKS
    - Using `eksctl` is our recommended option, as it creates a VPC and selects the proper security group for you.
  - Creating clusters - Google GKE
    - Do NOT use an Autopilot cluster. CircleCI requires functionality that is not supported by GKE Autopilot.

      Note that your Kubernetes cluster must meet the following minimum overall cluster requirements relative to the number of active CircleCI Server users:

Number of daily active CircleCI users	Minimum Nodes	Total CPU	Total RAM	NIC speed
< 500	3	8 cores	32 GB	1 Gbps
500+	3	48 cores	240 GB	10 Gbps

    - Your cluster must have outbound access to pull Docker containers and verify your license. If you do not want to provide open outbound access, see our list of ports that will need access.
- You must have appropriate permissions to list, create, edit and delete pods in your cluster. You can verify that you can list these resources by running: `kubectl auth can-i <list|create|edit|delete> pods`.
- A CircleCI License file. Contact CircleCI support for a license.
- The Required Tools tools installed
  - Note that the `kubectl` tool must also be configured to have access to your cluster.
    - Connect to Amazon EKS clusters - Amazon EKS
    - Configuring cluster access for kubectl - Google GKE
- Port access requirements are listed here:
  - Kubernetes Load Balancers
  - Kubernetes Nodes
  - Nomad Clients
  - External VMs

There are no requirements regarding VPC setup or disk size for your cluster. It is, however, recommended that you set up a new VPC rather than use an existing one.

Required Tools

Tool	Version	Used for
Terraform	{terraformversion} or greater	Infrastructure Management
kubectl	{kubectlversion} or greater	Kubernetes CLI
Helm	{helmversion} or greater	Kubernetes Package Management
Kots	{kotsversion} or greater	Replicated Kubernetes Application Management

External Ports

Port number	Protocol	Direction	Source / Destination	Use	Notes
80	TCP	Inbound	End users	HTTP web app traffic	
443	TCP	Inbound	End users	HTTP web app traffic	
8800	TCP	Inbound	Administrators	Admin console	
22	TCP	Inbound	Administrators	SSH	Only required for the bastion host
64535-65535	TCP	Inbound		SSH into builds	Only required for the nomad clients.

What to read next

- Creating your first cluster
- Server 3.x Installation

CircleCI Server v3.x - Creating Your First Cluster (Optional)

If you have never previously set up a Kubernetes cluster, we have provided some tips in this section.

Step 1 - Create

Amazon EKS

CircleCI recommends using `eksctl` to set up your first cluster on AWS. `eksctl` will take care of VPC creation, in addition to security group selection.

Before creating the cluster, make sure you have the following:

1. The latest AWS CLI installed and configured to your AWS account
2. `eksctl` installed
3. `kubectl` installed

Create a cluster using flags

To create a simple cluster, you can run the command:

```
eksctl create cluster
```

Additional flags are available on the command line. See the `eksctl` introduction for more information.

Create cluster using a configuration file

You can also create a config file for your cluster, for example:

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: <your-cluster-name>
  region: <aws-region>

managedNodeGroups:
  - name: <nodegroup-name-1>
    instanceType: <instance-type> # i.e., m5.large. see https://aws.amazon.com/ec2/instance-types/
for available instance types
    minSize: 4 # see https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-
as-group.html#cfn-as-group-minsize for more information
    maxSize: 6 # see https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-
as-group.html#cfn-as-group-maxsize for more information
```

For more examples on cluster configuration files, see eksctl.io.

When you are finished with your configuration, save it and run:

```
eksctl create cluster -f <your-cluster-config.yaml>
```

When using the `eksctl` tool to create your cluster, you may receive an AWS STS access error: `AWS STS access – cannot get role ARN for current session: InvalidClientTokenId.`

This may mean your AWS credentials are invalid, or your IAM user does not have permission to create an EKS cluster. Note that the proper IAM permissions are necessary in order to use `eksctl`. See the AWS documentation regarding prerequisite IAM permissions.

Step 2 - Verify

Once your cluster is finished being created, you should be able to run various `kubectl` commands to view your cluster resources.

For instance, to view your cluster's built-in services, you can run:

```
kubectl cluster-info
```

Or, to verify that your cluster has worker nodes attached, run:

```
kubectl get nodes -o wide
```

For more information on exploring your new cluster, see the following:

- Step 2: View resources - AWS docs
- Accessing clusters - Kubernetes docs
- Cluster Management - `kubectl` docs

What to read next

- Server 3.x Installation
- Hardening Your Cluster

include::server-3-install.adoc

CircleCI Server v3.x Migration

Migrating from 2.19.x to 3.0 requires that you backup your 2.19 installation and then restore the data to the new Server 3.0 installation. It does require an already operating Server 3.0 installation. Depending on the size of your data stores, the migration can take anywhere from a few minutes to a few hours. We recommend using a staging environment before completing this process in a production environment. This will not only allow you to gain a better understanding of the migration process, but will also give you a feel for how long the migration will take to complete.

If you have externalized your data stores (Mongo, Postgres, and Vault), then you need to wait until the Server 3.1 release for migration. Server 3.1 will support externalized data stores, and you will be able to point your 3.1 installation to the data stores you were using for your 2.19 instance.

Prerequisites

1. Your current CircleCI Server installation is 2.19
2. You have a running CircleCI Server 3.0 installation.
3. You have successfully run reality check with contexts prior to starting.
4. The migration script must be run from a machine with
    - `kubectl` configured for the Server 3.0 instance
    - `ssh` access to the 2.19 services box

Migration

Migrating to Server v3.x will shut down your v2.19.x application. Your v2.19.x application will not be started back up, although you may manually start it back up using the administrative console.

Starting the migration process will cause downtime. It is recommended you schedule a maintenance window.

Running Server 2.19 and Server 3.0 at the same time can cause issues to your 2.19 build data. Server 2.19 should NOT be restarted if Server 3.0 is running.

Step 1 - Clone the repository and run the migration script

The instructions below will clone the repository containing the Server v2.19.x to Server v3.x migration script. The migration script will:

- Stop your v2.19.x application
- Create a tarball of your v2.19.x application's PostgreSQL and Mongo databases
- Archive existing application data for Vault and CircleCI encryption/signing keys
- Export the tarball to your v3.x installation (exported data stores are stored in `migrate/circleci_export`,

which may be useful for debugging purposes).

- Scale v3.x application deployments down to zero, and then to one
- Import the data from the previously exported tarball to your new v3.x instance

In a terminal:

1. Run `git clone git@github.com:CircleCI-Public/server-scripts.git`
2. Change into the `migrate` directory: `cd server-scripts/migrate`
3. Run the migration script, i.e.: `./migrate.sh`
4. You will be prompted for the following information:
    - Username of your Server 2.19 installation
    - Hostname of your Server 2.19 installation
    - The path to your SSH key file for your Server 2.19 installation
    - Kubernetes namespace of your Server 3.0 installation
5. After the script has completed, the Signing and Encryption keys from the 2.19 instance will need to be added to the new 3.0 instance via the KOTS Admin Console. The keys will be located in `migrate/circleci_export/circle-data`
6. The 3.0 instance will either need to be updated to point at the same storage bucket that the 2.19 instance used, or the data needs to be copied over to a new bucket. The latter will ensure the 2.19 instance continues to work as expected, and so is the recommended approach if this migration is part of a test.

If a different hostname is being used in the 3.0 environment, the GitHub webhooks will still be pointing to the hostname used in the 2.19 environment. The easiest way to update this is to click **Stop Building** and then **Set Up Project**. After doing this, the contexts and environment variables associated with the project will still be present.

Step 2 - Validate your migration to Server 3.0

Re-run reality check with contexts on your new Server 3.0 environment by pushing a fresh commit.

Step 3 - Update your team

Once you have successfully run reality check, notify your team of the new CircleCI UI and URL, if it has changed.

Frequently Asked Questions

Where did all my job and build history go?

- All of your existing jobs and build history have been moved to the Legacy Jobs view. You can view the complete job history using one of the following methods:
    - Selecting Projects → PROJECT_NAME and selecting the `legacy jobs view` link at the bottom of the project's build history

- Using the following URL pattern: `https://<APP_DOMAIN>/pipelines/github/<ORG>/<PROJECT>/jobs`
- For specific job append a job number to the URL: <code><a href="https://<APP_DOMAIN>/pipelines/github/<ORG>/<PROJECT>/jobs/<JOB#&gt" class="bare">https://<APP_DOMAIN>/pipelines/github/<ORG>/<PROJECT>/jobs/<JOB#&gt;</code>

Why does nothing happen when I select "Start Building" on my project after migration?

- By default, a newly added project (a project that has never been followed) will trigger a build automatically after it has been followed for the first time. If the project was or ever has been followed in 2.0 or 3.0, it will not be considered a new project or first build and a build will not be triggered after follow. To trigger a build, perform an activity that will trigger a Github webhook such as pushing up a new commit or branch.

What to read next

- Hardening Your Cluster
- Server 3.x Operator Guide

CircleCI Server v3.x Hardening Your Cluster

This section provides supplemental information on hardening your Kubernetes cluster.

Network Topology

A server installation basically runs three different type of compute instances: The Kubernetes nodes, Nomad clients and external VMs.

It is highly recommended that you deploy these into separate subnets with distinct CIDR blocks. This will make it easier for you to control traffic between the different components of the system and isolate them from each other.

As always, the rule is to make as many of the resources private as possible, applies. If your users will access your CircleCI Server installation via VPN, there is no need to assign any public IP addresses at all, as long as you have a working NAT gateway setup. Otherwise, you will need at least one public subnet for the frontend load balancer.

However, in this case, it is also recommended to place Nomad clients and VMs in a public subnet to enable your users to SSH into jobs and scope access via networking rules.

Currently, custom subnetting is not supported for GCP. Custom subnetting support will be available in a future update/release.

Network Traffic

This section spells out the minimum requirements that are needed for a server installation to work. Depending on your workloads, you might need to add additional rules to egress for Nomad clients and VMs. Nomenclature between cloud providers differs, therefore, you will probably need to implement these rules using firewall rules and/or security groups.

Where you see "external," this usually means all external IPv4 addresses. Depending on your particular setup, you might be able to be more specific (e.g., if you are using a proxy for all external traffic).

It is assumed that you have configured the load balancers for Nomad, vm-service and output processor to be internal load balancers - this is the default.

The rules spelled out here are assumed to be stateful and for TCP connections only, unless stated otherwise. If you are working with stateless rules, you will need to create matching ingress or egress rules to the ones listed here.

Kubernetes Load Balancers

Depending on your setup, your load balancers might be transparent (i.e. they are not treated as a distinct layer in your networking topology). In this case, you can apply the rules from this section directly to the underlying destination or source of the network traffic. Refer to the documentation of your cloud provider to make sure you understand how to correctly apply networking security rules given the type of load balancing you are using with your installation.

Ingress

If the traffic rules for your load balancers have not been created automatically, here are their respective ports:

Name	Port	Source	Purpose
frontend-external	80	External	Frontend API
frontend-external	443	External	Frontend API
*-server-traefik	80	External	User Interface
*-server-traefik	443	External	User Interface
vm-service	3000	Nomad clients	Communication with Nomad clients
nomad	4647	Nomad clients	Communication with Nomad clients
output-processor	8585	Nomad clients	Communication with Nomad clients

Egress

The only type of egress needed is TCP traffic to the K8s nodes on the K8s load balancer ports (30000-32767). This is not needed if your load balancers are transparent.

Common Rules for Compute Instances

These rules apply to all compute instances, but not to the load balancers.

Ingress

If you want to access your instances via SSH, you will need to open port 22 for TCP connections for the instances in question. It is recommended to scope the rule as closely as possible to allowed source IPs and/or only add such a rule ad hoc when needed.

Egress

You most likely want all of your instances to access internet resources. This will require you to allow egress for UDP and TCP on port 53 to the DNS server within your VPC, as well as TCP ports 80 and 443 for HTTP and HTTPS traffic, respectively. Instances building jobs (i.e. the Nomad clients and external VMs) also will likely need to pull code from your VCS via SSH (TCP port 22). SSH is also used to communicate with external VMs, so it should be allowed for all instances with the destination of the VM subnet and your VCS at the very least.

Kubernetes Nodes

Intra-node traffic

The traffic within your K8s cluster is regulated by networking policies by default. For most purposes, this should be sufficient to regulate the traffic between pods and there is no additional requirement to pare down traffic between K8s nodes any further (i.e. it is fine to allow all traffic between K8s nodes).

To make use of networking policies within your cluster, you may need to take additional steps, depending on your cloud provider and setup. Here are some resources to get you started:

- Kuberenetes Network Policy Overview
- Creating a Cluster Network Policy on Google Cloud
- Installing Calico on Amazon EKS

Ingress

If you are using a managed service, you can check the rules created for the traffic coming from the load balancers and the allowed port range. The standard port range for K8s load balancers (30000-32767) should be all that is needed here for ingress. If you are using transparent load balancers, you will need to apply the ingress rules listed for load balancers above.

Egress

Port	Destination	Purpose
2376	VMs	Communication with VMs
4647	Nomad clients	Communication with the Nomad clients
all traffic	other nodes	Allow intra-cluster traffic

Nomad Clients

Nomad clients do not need to communicate with each other; you can block traffic between Nomad client instances completely.

Ingress

Port	Source	Purpose
4647	K8s nodes	Communication with Nomad server
64535-65535	External	Rerun jobs with SSH functionality

Egress

Port	Destination	Purpose
2376	VMs	Communication with VMs

Port	Destination	Purpose
3000	VM Service load balancers	Internal communication
4647	Nomad Load Balancer	Internal communication
8585	Output Processor Load Balancer	Internal communication

External VMs

Similar to Nomad clients, there is no need for external VMs to communicate with each other.

Ingress

Port	Source	Purpose
22	K8s nodes	Internal communication
22	Nomad clients	Internal communication
2376	K8s nodes	Internal communication
2376	Nomad clients	Internal communication
64535-65535	External	Rerun jobs with SSH functionality

Egress

You will only need the egress rules for internet access and SSH for your VCS.

What to read next

- Server 3.x Migration
- Server 3.x Operations