CircleCI Server v2.16 Installation Guide

Final Documentation

April 3rd, 2019
## Contents

**Overview**
- Build Environments ........................................ 5
- Architecture .................................................. 5
- Services Instance ............................................. 6
- Nomad Clients ................................................ 7
- GitHub .......................................................... 7

**Installing CircleCI v2.16 on Amazon Web Services with Terraform**
- Externalization ............................................... 9
- Installation Prerequisites .................................. 10
- Private Subnet Requirements ............................... 10
- Planning ....................................................... 11
- Installation with Terraform ............................... 12
- Validating your Installation ............................... 13
- Troubleshooting ............................................. 13

**Troubleshooting**
- FAQ .......................................................... 15
  - Can I move or change my GitHub Enterprise URL without downtime? ........................................ 15
  - Can I monitor available build containers? .................. 15
  - How do I provision admin users? .......................... 15
  - How can I gracefully shutdown Nomad Clients? .... 16
  - Why is Test GitHub Authentication failing? ............ 16
  - How can I use HTTPS to access CircleCI? ............. 16
  - Why doesn’t terraform destroy every resource? .... 16
  - Do the Nomad Clients store any state? ................ 16
  - How do I verify TLS settings are failing? ............ 16
  - How do I debug the Management Console (Replicated)? .... 17

**Server Ports** .................................................. 25
Overview

CircleCI is a modern continuous integration and continuous delivery (CI/CD) platform installable inside your private cloud or data center.

CircleCI 2.x provides new infrastructure that includes the following improvements:

- New configuration with any number of jobs and workflows to orchestrate them.
- Custom images for execution on a per-job basis.
- Fine-grained performance with custom caching and per-job CPU or memory allocation.

Refer to the v2.16 Changelog at https://circleci.com/server/changelog/#2-16-00 for what’s new in this release.

Build Environments

CircleCI uses Nomad as the primary job scheduler in CircleCI 2.x. Refer to the Introduction to Nomad Cluster Operation to learn more about the job scheduler and how to perform basic client and cluster operations.

By default, CircleCI 2.x Nomad clients automatically provision containers according to the image configured for each job in your .circleci/config.yml file.

Architecture

Figure 1.1 illustrates CircleCI core components, build orchestration services, and executors. The CircleCI API is a full-featured RESTful API that allows you to access all information and trigger all actions in CircleCI. The Insights page in the CircleCI UI is a dashboard showing the health of all repositories you are following.
including median build time, median queue time, last build time, success rate, and parallelism.

Figure 1: CircleCI Services Architecture

CircleCI consists of two primary components: Services and Nomad Clients. Any number of Nomad Clients execute your jobs and communicate back to the Services. All components must access GitHub or your hosted instance of GitHub Enterprise on the network as illustrated in the following architecture diagram.

**Services Instance**

The machine on which the Services instance runs must not be restarted and may be backed up using VM snapshotting. If you must restart the Services machine, do so only as a last resort because restart will result in downtime. Refer to the Disaster Recovery chapter for instructions.

DNS resolution may point to the IP address of the machine on which the Services are installed. It is also possible to point to a load balancer, like for example an ELB in AWS. The following table describes the ports used for traffic on the Service instance:

<table>
<thead>
<tr>
<th>Source</th>
<th>Ports</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>End Users</td>
<td>80, 443, 4434</td>
<td>HTTP/HTTPS Traffic</td>
</tr>
<tr>
<td>Administrators</td>
<td>22</td>
<td>SSH</td>
</tr>
</tbody>
</table>
Nomad Clients

The Nomad Clients run without storing state, enabling you to increase or decrease the number of containers as needed.

To ensure that there are enough running to handle all of the builds, track the queued builds and increase the number of Nomad Client machines as needed to balance the load.

Each machine reserves two CPUs and 4GB of memory for coordinating builds. The remaining processors and memory create the containers. Larger machines are able to run more containers and are limited by the number of available cores after two are reserved for coordination.

**Note:** The maximum machine size for a Nomad client is 128GB RAM/64 CPUs, contact your CircleCI account representative to request use of larger machines for Nomad Clients.

The following table describes the ports used on the Nomad clients:

<table>
<thead>
<tr>
<th>Source</th>
<th>Ports</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>End Users</td>
<td>64535-65535</td>
<td>SSH into builds</td>
</tr>
<tr>
<td>Administrators</td>
<td>80 or 443</td>
<td>CCI API Access</td>
</tr>
<tr>
<td>Administrators</td>
<td>22</td>
<td>SSH</td>
</tr>
<tr>
<td>Services Machine</td>
<td>all traffic / all ports</td>
<td>Internal Commns</td>
</tr>
<tr>
<td>Nomad Clients (including itself)</td>
<td>all traffic / all ports</td>
<td>Internal Commns</td>
</tr>
</tbody>
</table>

GitHub

CircleCI uses GitHub or GitHub Enterprise credentials for authentication which, in turn, may use LDAP, SAML, or SSH for access. That is, CircleCI will inherit the authentication supported by your central SSO infrastructure. **Note:** CircleCI does not support changing the URL or backend Github instance after it has been set up. The following table describes the ports used on machines running GitHub to communicate with the Services and Builder instances.
Figure 2: A Diagram of the CircleCI Architecture

<table>
<thead>
<tr>
<th>Source</th>
<th>Ports</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Services</td>
<td>22</td>
<td>Git Access</td>
</tr>
<tr>
<td>Services</td>
<td>80, 443</td>
<td>API Access</td>
</tr>
<tr>
<td>Nomad Client</td>
<td>22</td>
<td>Git Access</td>
</tr>
<tr>
<td>Nomad Client</td>
<td>80, 443</td>
<td>API Access</td>
</tr>
</tbody>
</table>
Installing CircleCI v2.16 on Amazon Web Services with Terraform

This document provides step-by-step instructions for installing CircleCI v2.16 on Amazon Web Services (AWS) with Terraform in the following sections. Refer to https://circleci.com/server/changelog for what's new and fixed in this release.

Notes: - CircleCI 2.0 may be installed without a support agreement on AWS using the examples and instructions in this document. - It is possible to install and configure CircleCI on Azure or any other platform used in your organization with a Platinum CircleCI support agreement. Contact CircleCI support or your account representative to get started.

Externalization

With a Platinum support agreement, it is possible to configure the following services to run external to the Services machine for improved performance:

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

Contact support to evaluate your installation against the current requirements for running external services.
Installation Prerequisites

Install the following automated infrastructure provisioning software:

- Terraform, see the Download Terraform web site for links to packages for your architecture.

Have the following information available before beginning the installation procedure:

- CircleCI License file (.rli), contact CircleCI support for a license.
- AWS Access Key, AWS Secret Key.
- Name of AWS EC2 SSH key.
- AWS Region, for example us-west-2.
- AWS Virtual Private Cloud (VPC) ID and AWS Subnet ID. Your default VPC ID is listed under Account Attributes in Amazon if your account is configured to use a default VPC.
- 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. **Note:** 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 section of the Administrator's 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 Preview Release installation:

• If you use network proxies, contact your Account team before attempting to install CircleCI 2.0.
• Plan to provision at least two AWS instances, one for the 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 the Services as well as Nomad Clients instances.
• AWS instances must have outbound access to pull Docker containers and to verify your license.
• In order to provision required AWS entities with Terraform you need an IAM User with following permissions:

```json
{
    "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"
            ]
        }
    ]
}
```
Installation with 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: `git checkout master && git pull`).
2. Run `make init` to init `terraform.tfvars` file (your previous `terraform.tfvars` if any, will be backed up in the same directory).
3. Fill `terraform.tfvars` with appropriate AWS values for section 1.
4. Specify a `circle_secret_passphrase` in section 2, replacing ... with alpha numeric characters. Passphrase cannot be empty.
5. Specify the instance type 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. **Note:** The `builder_instance_type` is only used for 1.0 and is disabled by default in section 3.
6. Run `terraform apply` to provision.
Validating your Installation

1. Click the Open link in the dashboard to go to the CircleCI app. The Starting page appears for a few minutes as the CircleCI application is booting up, then automatically redirects to the homepage.
2. Sign up or sign in by clicking the Get Started button. Because you are the first user to log in, you become the Administrator.
3. Add a project using the Hello World document.

Troubleshooting

If you’re unable to run your first builds successfully please start with the Operations Guide.
After the build containers start and complete downloading of images, the first build should begin immediately.

If there are no updates after about 15 minutes and you have clicked the Refresh button, contact CircleCI support for assistance.
Troubleshooting

This chapter answers frequently asked questions and provides installation troubleshooting tips.

FAQ

Can I move or change my GitHub Enterprise URL without downtime?

No, because of the nature of CircleCI integration with GitHub authentication, you should not change the domain of your GHE instance after CircleCI is in production. Redeploying GitHub without will result in a corrupted CircleCI instance. Contact support if you plan to move your GitHub instance.

Can I monitor available build containers?

Yes, refer to the Introduction to Nomad Cluster Operation document for details. Refer to the Administrative Variables, Monitoring, and Logging section for how to enable additional container monitoring for AWS.

How do I provision admin users?

The first user who logs in to the CircleCI application will automatically be designated an admin user. Options for designating additional admin users are found under the Users page in the Admin section at https://[domain-to-your-installation]/admin/users.
How can I gracefully shutdown Nomad Clients?

Refer to the Introduction to Nomad Cluster Operation chapter for details.

Why is Test GitHub Authentication failing?

This means that the GitHub Enterprise server is not returning the intermediate SSL certificates. Check your GitHub Enterprise instance with https://www.ssllabs.com/ssltest/analyze.html - it may report some missing intermediate certs. You can use commands like openssl to get the full certificate chain for your server.

In some cases authentication fails when returning to the configuration page after it was successfully set up once. This is because the secret is encrypted, so when returning checking it will fail.

How can I use HTTPS to access CircleCI?

While CircleCI creates a self-signed cert when starting up, that certificate only applies to the management console and not the CircleCI product itself. If you want to use HTTPS, you'll have to provide certificates to use under the Privacy section of the settings in the management console.

Why doesn’t terraform destroy every resource?

CircleCI sets the services box to have termination protection in AWS and also writes to an s3 bucket. If you want terraform to destroy every resource, you'll have to either manually delete the instance, or turn off termination protection in the circleci.tf file. You'll also need to empty the s3 bucket that was created as part of the terraform install.

Do the Nomad Clients store any state?

They can be torn down without worry as they don't persist any data.

How do I verify TLS settings are failing?

Make sure that your keys are in unencrypted PEM format, and that the certificate includes the entire chain of trust as follows:
How do I debug the Management Console (Replicated)?

If you’re experiencing any issues with Replicated, here are a few ways to debug it.

Check the current version of Replicated installed

First, make sure you have the CLI tool for Replicated installed:

```
replicated -version
```

Restart Replicated and the CircleCI app

Try restarting Replicated services. You can do this by running the following commands on the service box for Ubuntu 14.04:

```
sudo restart replicated-ui
sudo restart replicated
sudo restart replicated-agent
```

For Ubuntu 16.04, run the following commands:

```
sudo systemctl restart replicated-ui
sudo systemctl restart replicated
sudo systemctl restart replicated-operator
```

Then, go to your services box admin (for example, https://YOUR-CCIE-INSTALL:8800) and try restarting with “Stop Now” and “Start Now”.

Try to log into Replicated

Try to log in to Replicated. You can do this by running the following commands on the service box. You will only be asked to enter password, which is the same
one used to unlock the admin (i.e.: https://YOUR-CCIE-INSTALL:8800).

replicated login

If you could login, then please run the following command too and give us the output.

```
sudo replicated apps
```

You are getting error: request returned Unauthorized for API route. Error probably because you are not logged into Replicated, so please check if you are still getting the error after successful login.

**Check Replicated logs**

You can find Replicated logs under `/var/log/replicated`.

**Check output of docker ps**

Replicated starts many Docker containers to run CCIE, so it may be useful to check what containers are running.

You should see something similar to this output:

```
sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
eb2970306859 172.31.72.162:9874/circleci-api-service:0.1.6910-8b54ef9 "circleci-service-run" 26 hours ago Up 26 hours 0.0.0.0:32872->80/tcp, 0.0.0.0:32871->443/tcp, 0.0.0.0:32870->6010/tcp, 0.0.0.0:32869->8585/tcp api-service
01d26714f5f5 172.31.72.162:9874/circleci-workflows-conductor:0.1.38931-1a904bc8 "service/docker-ent..." 26 hours ago Up 26 hours 0.0.0.0:9998->9998/tcp, 0.0.0.0:32868->80/tcp, 0.0.0.0:32867->443/tcp, 0.0.0.0:32866->3000/tcp, 0.0.0.0:32866->8585/tcp workflows-conductor
0cc6e428c9f5 172.31.72.162:9874/circleci-permissions-service:0.1.1195-b617002 "service/docker-ent..." 26 hours
```
<table>
<thead>
<tr>
<th>Service Name</th>
<th>Image</th>
<th>Port Range</th>
<th>Status</th>
<th>Logs Age</th>
<th>IP Address</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>0.1.1370-ad05648</td>
<td>0.0.0.0:3001-&gt;3000/tcp</td>
<td>Up 26 hours</td>
<td>26 hours ago</td>
<td>172.31.72.162:9874/circleci</td>
<td>service-....</td>
</tr>
<tr>
<td>vm-service</td>
<td>0.1.1370-ad05648</td>
<td>0.0.0.0:3001-&gt;3000/tcp</td>
<td>Up 26 hours</td>
<td>26 hours ago</td>
<td>172.31.72.162:9874/circleci</td>
<td>service-....</td>
</tr>
<tr>
<td>vm-scaler</td>
<td>0.1.1370-ad05648</td>
<td>0.0.0.0:3001-&gt;3000/tcp</td>
<td>Up 26 hours</td>
<td>26 hours ago</td>
<td>172.31.72.162:9874/circleci</td>
<td>vm-scaler:....</td>
</tr>
<tr>
<td>vm-scaler:entrypoint</td>
<td>0.1.1370-ad05648</td>
<td>0.0.0.0:3001-&gt;3000/tcp</td>
<td>Up 26 hours</td>
<td>26 hours ago</td>
<td>172.31.72.162:9874/circleci</td>
<td>vm-scaler:....</td>
</tr>
<tr>
<td>vm-gc</td>
<td>0.1.1370-ad05648</td>
<td>0.0.0.0:3001-&gt;3000/tcp</td>
<td>Up 26 hours</td>
<td>26 hours ago</td>
<td>172.31.72.162:9874/circleci</td>
<td>vm-gc:....</td>
</tr>
<tr>
<td>vm-gc:entrypoint</td>
<td>0.1.1370-ad05648</td>
<td>0.0.0.0:3001-&gt;3000/tcp</td>
<td>Up 26 hours</td>
<td>26 hours ago</td>
<td>172.31.72.162:9874/circleci</td>
<td>vm-gc:....</td>
</tr>
<tr>
<td>output-processing</td>
<td>1.1.10386-741e1d1</td>
<td>0.0.0.0:32863-&gt;80/tcp, 0.0.0.0:32862-&gt;443/tcp</td>
<td>Up 26 hours</td>
<td>26 hours ago</td>
<td>172.31.72.162:9874/circleci</td>
<td>output-processing:....</td>
</tr>
<tr>
<td>output-processor</td>
<td>1.1.10386-741e1d1</td>
<td>0.0.0.0:32863-&gt;80/tcp, 0.0.0.0:32862-&gt;443/tcp</td>
<td>Up 26 hours</td>
<td>26 hours ago</td>
<td>172.31.72.162:9874/circleci</td>
<td>output-processor:....</td>
</tr>
<tr>
<td>frontend</td>
<td>0.1.203321-501fada</td>
<td>0.0.0.0:32861-&gt;80/tcp, 0.0.0.0:32860-&gt;443/tcp, 0.0.0.0:32859-&gt;4434/tcp</td>
<td>Up 26 hours</td>
<td>26 hours ago</td>
<td>172.31.72.162:9874/circleci</td>
<td>frontend:....</td>
</tr>
<tr>
<td>frontend:entrypoint</td>
<td>0.1.203321-501fada</td>
<td>0.0.0.0:32861-&gt;80/tcp, 0.0.0.0:32860-&gt;443/tcp, 0.0.0.0:32859-&gt;4434/tcp</td>
<td>Up 26 hours</td>
<td>26 hours ago</td>
<td>172.31.72.162:9874/circleci</td>
<td>frontend:....</td>
</tr>
<tr>
<td>legacy-notifier</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>legacy-notifier</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mongo</td>
<td>3.6.6-jessie</td>
<td>0.0.0.0:27017/tcp</td>
<td>Up 26 hours</td>
<td>26 hours ago</td>
<td>172.31.72.162:9874/mongo:3.6-jessie</td>
<td>mongo:3.6.6-jessie</td>
</tr>
<tr>
<td>mongo:entrypoint</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>telegraf</td>
<td>1.6.4</td>
<td>0.0.0.0:8125/udp, 0.0.0.0:32771-&gt;8092/udp, 0.0.0.0:32855-&gt;8094/tcp</td>
<td>Up 26 hours</td>
<td>26 hours ago</td>
<td>172.31.72.162:9874/telegraf:1.6.4</td>
<td>telegraf:1.6.4</td>
</tr>
<tr>
<td>telegraf:entrypoint</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>circleci-schedulerer</td>
<td>0.1.10388-741e1d1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>circleci-schedulerer:entrypoint</td>
<td>0.1.10388-741e1d1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FAQ

```
service-run" 26 hours ago  Up 26 hours
scheduler
fb967bd3bca0  172.31.72.162:9874/circleci-
server-nomad:0.5.6-5.1
"/nomad-
entrypoint.sh" 26 hours ago  Up 26 hours  0.0.0.0:4646-4648->4646-
4648/tcp
7e0743ee2bfc  172.31.72.162:9874/circleci-
test-results:0.1.1136-b4d94f6
"circleci-
service-run" 26 hours ago  Up 26 hours  0.0.0.0:2719-
>2719/tcp, 0.0.0.0:3012->3012/tcp
test-
results
0a95802c87dc  172.31.72.162:9874/circleci-
slanger:0.4.117-42f7e6c
"/docker-
entrypoint..." 26 hours ago  Up 26 hours  0.0.0.0:4567-
>4567/tcp, 0.0.0.0:8081->8080/tcp
c445870a057  172.31.72.162:9874/circleci-
postgres-script-enhance:0.1.9-38edabf
"docker-
entrypoint.s..." 26 hours ago  Up 26 hours  0.0.0.0:5432-
>5432/tcp
a563a228a93a  172.31.72.162:9874/circleci-
server-ready-agent:0.1.105-0193c73
"/server-
ready-agent" 26 hours ago  Up 26 hours  0.0.0.0:8099-
>8000/tcp
agent
d6f9aaaee5cf2  172.31.72.162:9874/circleci-
server-usage-stats:0.1.122-70f28aa
"bash -c /src/entryp..." 26 hours ago  Up 26 hours
stats
086a53d91a1a5  registry.replicated.com/library/statsd-
graphite:0.3.7
"/usr/bin/supervisor..." 26 hours ago  Up 26 hours  0.0.0.0:32851-
>2443/tcp, 0.0.0.0:32770->8125/udp
statsd
c55e062844be  172.31.72.162:9874/circleci-
shutdown-hook-poller:0.1.32-9c553b4
"/usr/local/bin/pyth..." 26 hours ago  Up 26 hours
rabbitmq-delayed:3.6.6-management-12
"docker-
entrypoint.s..." 26 hours
```
TROUBLESHOOTING

ago Up 26 hours 0.0.0.0:5672->5672/tcp, 0.0.0.0:15672->15672/tcp, 0.0.0.0:32850->4369/tcp, 0.0.0.0:32849->5671/tcp, 0.0.0.0:32848->15671/tcp, 0.0.0.0:32847->25672/tcp rabbitmq
2bc0cfe43639 172.31.72.162:9874/tutum-logrotate:latest "crond -f" 26 hours
ago Up 26 hours 79aa857e23b4 172.31.72.162:9874/circleci-vault-cci:0.3.8-e2823f6 "./docker-entrypoint.sh" 26 hours
ago Up 26 hours 172.31.72.162:9874/circleci-cci:0.3.8-e2823f6/.docker-entrypoint.sh 26 hours
ago Up 26 hours 0.0.0.0:8200-8201->8200-8201/tcp cci
b3e317c9d62f 172.31.72.162:9874/redis:4.0.10 entrypoint.sh" 26 hours
ago Up 26 hours 0.0.0.0:6379->6379/tcp
f2d3f77891f0 172.31.72.162:9874/circleci-nomad-metrics:0.1.90-1448fa7 "/usr/local/bin/dock" 26 hours
ago Up 26 hours metrics
1947a7038f24 172.31.72.162:9874/redis:4.0.10 entrypoint.sh" 26 hours
ago Up 26 hours 0.0.0.0:32846->6379/tcp
redis
3899237a5782 172.31.72.162:9874/circleci-exim:0.2.54-697cd08 "/docker-entrypoint.sh" 26 hours
ago Up 26 hours 0.0.0.0:2525->25/tcp
reddis
97ebdb831a7e registry.replicated.com/library/retraced:1.2.2 audit.sh" 26 hours
ago Up 26 hours 3000/tcp processor
a0b806f3fad2 registry.replicated.com/library/retraced:1.2.2 audit.sh" 26 hours
ago Up 26 hours 172.17.0.1:32771->3000/tcp
api
19dec5045f6e registry.replicated.com/library/retraced:1.2.2 /usr/loc..." 26 hours
ago Up 26 hours 3000/tcp
cron
7b83a3a193da registry.replicated.com/library/retraced-
Providing support with the output of `sudo docker ps` in service box will be helpful in diagnosing the problem.
Server Ports

This chapter provides System Administrators with a complete list of ports for the machines in their CircleCI 2.0 installation:

<table>
<thead>
<tr>
<th>Machine type</th>
<th>Port number</th>
<th>Protocol</th>
<th>Direction</th>
<th>Source / destination</th>
<th>Use</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Services Machine</td>
<td>80</td>
<td>TCP</td>
<td>Inbound</td>
<td>End users</td>
<td>HTTP web app traffic</td>
<td></td>
</tr>
<tr>
<td>Services Machine</td>
<td>443</td>
<td>TCP</td>
<td>Inbound</td>
<td>End users</td>
<td>HTTPS web app traffic</td>
<td></td>
</tr>
<tr>
<td>Services Machine</td>
<td>7171</td>
<td>TCP</td>
<td>Inbound</td>
<td>End users</td>
<td>Artifacts access</td>
<td></td>
</tr>
<tr>
<td>Services Machine</td>
<td>8081</td>
<td>TCP</td>
<td>Inbound</td>
<td>End users</td>
<td>Artifacts access</td>
<td></td>
</tr>
<tr>
<td>Services Machine</td>
<td>22</td>
<td>TCP</td>
<td>Inbound</td>
<td>Administrators</td>
<td>Ssh admin console</td>
<td></td>
</tr>
<tr>
<td>Services Machine</td>
<td>8800</td>
<td>TCP</td>
<td>Inbound</td>
<td>Administrators</td>
<td>Metrics</td>
<td></td>
</tr>
<tr>
<td>Database Servers</td>
<td>8125</td>
<td>UDP</td>
<td>Inbound</td>
<td>Nomad Clients</td>
<td>Metrics</td>
<td>Only if using externalised databases</td>
</tr>
<tr>
<td>Database Servers</td>
<td>8125</td>
<td>UDP</td>
<td>Inbound</td>
<td>Nomad Servers</td>
<td>Metrics</td>
<td>Only if using externalised databases</td>
</tr>
<tr>
<td>Database Servers</td>
<td>8125</td>
<td>UDP</td>
<td>Inbound</td>
<td>All Database Servers</td>
<td>Metrics</td>
<td>Only if using externalised databases</td>
</tr>
<tr>
<td>Machine type</td>
<td>Port number</td>
<td>Protocol</td>
<td>Direction</td>
<td>Source / destination</td>
<td>Use</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td>4647</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>Nomad Clients</td>
<td>Internal communication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8585</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>Nomad Clients</td>
<td>Internal communication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7171</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>Nomad Clients</td>
<td>Internal communication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3001</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>Nomad Clients</td>
<td>Internal communication</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>GitHub Enterprise / GitHub.com (whichever applies)</td>
<td>Webhooks / API access</td>
<td></td>
</tr>
<tr>
<td></td>
<td>443</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>GitHub Enterprise / GitHub.com (whichever applies)</td>
<td>Webhooks / API access</td>
<td></td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>TCP</td>
<td>Outbound</td>
<td>AWS API endpoints</td>
<td>API access</td>
<td>Only if running on AWS</td>
</tr>
<tr>
<td></td>
<td>443</td>
<td>TCP</td>
<td>Outbound</td>
<td>AWS API endpoints</td>
<td>API access</td>
<td>Only if running on AWS</td>
</tr>
<tr>
<td></td>
<td>5432</td>
<td>TCP</td>
<td>Outbound</td>
<td>PostgreSQL Servers</td>
<td>PostgreSQL database connection</td>
<td>Only if using externalised databases. Port is user-defined, assuming the default PostgreSQL port.</td>
</tr>
<tr>
<td>Machine type</td>
<td>Port number</td>
<td>Protocol</td>
<td>Direction</td>
<td>Source / destination</td>
<td>Use</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td><strong>27017</strong></td>
<td>TCP</td>
<td>Outbound</td>
<td>MongoDB Servers</td>
<td>MongoDB database connection</td>
<td>Only if using externalised databases. Port is user-defined, assuming the default MongoDB port.</td>
<td></td>
</tr>
<tr>
<td><strong>5672</strong></td>
<td>TCP</td>
<td>Outbound</td>
<td>RabbitMQ Servers</td>
<td>RabbitMQ connection</td>
<td>Only if using externalised RabbitMQ.</td>
<td></td>
</tr>
<tr>
<td><strong>6379</strong></td>
<td>TCP</td>
<td>Outbound</td>
<td>Redis Servers</td>
<td>Redis connection</td>
<td>Only if using externalised Redis.</td>
<td></td>
</tr>
<tr>
<td><strong>4647</strong></td>
<td>TCP</td>
<td>Outbound</td>
<td>Nomad Servers</td>
<td>Nomad Server connection</td>
<td>Only if using externalised Nomad Servers.</td>
<td></td>
</tr>
<tr>
<td><strong>443</strong></td>
<td>TCP</td>
<td>Outbound</td>
<td>CloudWatch Metrics Endpoints</td>
<td>Only if using AWS CloudWatch.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Nomad Clients</strong></td>
<td>64535-65535</td>
<td>TCP</td>
<td>Inbound</td>
<td>End users</td>
<td>SSH into builds feature CircleCI Admin API access</td>
<td></td>
</tr>
<tr>
<td>Machine type</td>
<td>Port number</td>
<td>Protocol</td>
<td>Direction</td>
<td>Source / destination</td>
<td>Use</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>443</td>
<td>TCP</td>
<td>Inbound</td>
<td>Administrators CircleCI Admin API access SSHP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>TCP</td>
<td>Inbound</td>
<td>Administrators SSHP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>TCP</td>
<td>Outbound</td>
<td>GitHub Enterprise / GitHub.com Github. (whichever applies)</td>
<td>Download Code</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TCP</td>
<td>Bi-directional</td>
<td>Services Machine Internal communication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8585</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>Services Machine Internal communication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7171</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>Services Machine Internal communication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3001</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>Services Machine Internal communication</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>443</td>
<td>TCP</td>
<td>Outbound</td>
<td>Cloud Storage Provider Artifacts storage</td>
<td>Only if using external artifacts storage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>UDP</td>
<td>Outbound</td>
<td>Internal DNS Server DNS resolution</td>
<td>This is to make sure that your jobs can resolve all DNS names that are needed for their correct operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Machine type</td>
<td>Port number</td>
<td>Protocol</td>
<td>Direction</td>
<td>Source / destination</td>
<td>Use</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>GitHub Enterprise / GitHub.com (whichever applies)</td>
<td>22</td>
<td>TCP</td>
<td>Inbound</td>
<td>Services Machine</td>
<td>Git access</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>TCP</td>
<td>Inbound</td>
<td>Nomad Clients</td>
<td>Git access</td>
<td></td>
<td></td>
</tr>
<tr>
<td>443</td>
<td>TCP</td>
<td>Inbound</td>
<td>Nomad Clients</td>
<td>API access</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>Nomad Clients</td>
<td>API access</td>
<td></td>
<td></td>
</tr>
<tr>
<td>443</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>Services Machine</td>
<td>Webhooks / API access</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PostgreSQL Servers</td>
<td>5432</td>
<td>TCP</td>
<td>Inbound</td>
<td>Services Machine</td>
<td>PostgreSQL database connection</td>
<td>Only if using externalised databases. Port is user-defined, assuming the default PostgreSQL port.</td>
</tr>
<tr>
<td>Machine type</td>
<td>Port number</td>
<td>Protocol</td>
<td>Direction</td>
<td>Source / destination</td>
<td>Use</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>PostgreSQL 5432</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>PostgreSQL Servers</td>
<td>PostgreSQL replication</td>
<td>Only if using externalised databases. Port is user-defined, assuming the default PostgreSQL port.</td>
<td></td>
</tr>
<tr>
<td>MongoDB servers 27017</td>
<td>TCP</td>
<td>Inbound</td>
<td>Services Machine</td>
<td>MongoDB database connection</td>
<td>Only if using externalised databases. Port is user-defined, assuming the default MongoDB port.</td>
<td></td>
</tr>
<tr>
<td>27017</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>MongoDB Servers</td>
<td>MongoDB replication</td>
<td>Only if using externalised databases. Port is user-defined, assuming the default MongoDB port.</td>
<td></td>
</tr>
<tr>
<td>Machine type</td>
<td>Port number</td>
<td>Protocol</td>
<td>Direction</td>
<td>Source / destination</td>
<td>Use</td>
<td>Notes</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>----------</td>
<td>-----------</td>
<td>----------------------</td>
<td>----------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>RabbitMQ Servers</td>
<td>5672</td>
<td>TCP</td>
<td>Inbound</td>
<td>Services Machine</td>
<td>RabbitMQ connection</td>
<td>Only if using externalised RabbitMQ</td>
</tr>
<tr>
<td></td>
<td>5672</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>RabbitMQ Servers</td>
<td>RabbitMQ mirroring</td>
<td>Only if using externalised RabbitMQ</td>
</tr>
<tr>
<td>Redis Servers</td>
<td>6379</td>
<td>TCP</td>
<td>Inbound</td>
<td>Services Machine</td>
<td>Redis connection</td>
<td>Only if using externalised Redis</td>
</tr>
<tr>
<td></td>
<td>6379</td>
<td>TCP</td>
<td>Bi-directional</td>
<td>Redis Servers</td>
<td>Redis replication</td>
<td>Only if using externalised Redis and using Redis replication (optional)</td>
</tr>
<tr>
<td>Nomad Servers</td>
<td>4646</td>
<td>TCP</td>
<td>Inbound</td>
<td>Services Machine</td>
<td>Nomad Server connection</td>
<td>Only if using externalised Nomad Servers</td>
</tr>
<tr>
<td></td>
<td>4647</td>
<td>TCP</td>
<td>Inbound</td>
<td>Services Machine</td>
<td>Nomad Server connection</td>
<td>Only if using externalised Nomad Servers</td>
</tr>
<tr>
<td>Machine type</td>
<td>Port number</td>
<td>Protocol</td>
<td>Direction</td>
<td>Source / destination</td>
<td>Use</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
<td>----------</td>
<td>---------------</td>
<td>----------------------</td>
<td>------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>4648</td>
<td>TCP</td>
<td>Bi-directional</td>
<td><strong>Nomad Servers</strong></td>
<td><strong>Nomad Servers internal communication</strong></td>
<td>Only if using externalised Nomad Servers</td>
</tr>
</tbody>
</table>