Automate High availability

[HA cluster introduction 2](#_Toc1735579699)

[HA Components 3](#_Toc1457453763)

[High availability Architecture 4](#_Toc352595509)

[Reference Architecture 4](#_Toc1674402493)

[Automate Backend Cluster 4](#_Toc470756001)

[Automate Frontend Cluster 4](#_Toc370139149)

[Deployment Support Types 5](#_Toc1300465887)

[AWS Deployment 5](#_Toc23116399)

[Bare Infrastructure Deployment 5](#_Toc14734623)

[System and software requirements 5](#_Toc2051477031)

[Platform support 5](#_Toc1071771892)

[Virtual Machine (VM) Instances Type 6](#_Toc1422389617)

[Bastion host 6](#_Toc857065517)

[Bastion Introduction 6](#_Toc864442506)

[Bastion Host Setup 6](#_Toc1074597171)

[Getting started 7](#_Toc1878638185)

[Package download 7](#_Toc1992039151)

[Configuration and Provisioning – Cloud 7](#_Toc1755130240)

[Cloud System Requirements 7](#_Toc206866167)

[Virtual Machine (VM) Instances Type 7](#_Toc1463986513)

[Amazon’s Virtual Private Cloud (VPC) 8](#_Toc2071204290)

[VPC requirements 8](#_Toc1012289915)

[Amazon’s Virtual Private Cloud (VPC) Limit 8](#_Toc821219597)

[Configuration￼ 9](#_Toc672637901)

[Provisioning 9](#_Toc1402858549)

[AWS provisioning 9](#_Toc322668150)

[On-prem Configuration 11](#_Toc1675936222)

[On-prem Prerequisite 11](#_Toc1901740272)

[Configuration ￼ ￼ 11](#_Toc2023204849)

[Validation 13](#_Toc1393399874)

[Installation 13](#_Toc1753984586)

[Air-gapped installation 13](#_Toc263100213)

[Backup and restore 13](#_Toc881996409)

[Pre-back-up configuration: 13](#_Toc1731959389)

[ES configuration and setup 14](#_Toc675192601)

[S3 Configuration for backup 15](#_Toc1381317600)

[File System (EFS)Configuration for backup 18](#_Toc266541933)

[Backup 18](#_Toc1216523958)

[Restore 18](#_Toc207232882)

[Upgrade 18](#_Toc444249810)

[Migration 18](#_Toc1247082504)

[Chef server (HA- backend) to Automate HA 18](#_Toc1626081780)

[Backup on your existing chef-server: 19](#_Toc51881102)

[Restore to Chef Automate HA chef-server 20](#_Toc174870704)

[Existing A2HA to Automate HA 20](#_Toc2055014839)

[A2 to Automate HA 21](#_Toc1188665617)

[Performance benchmarking 21](#_Toc1562555762)

[Certificates renewal 22](#_Toc630760781)

[Security and firewall 24](#_Toc781351583)

[Incoming frontends network traffic 24](#_Toc662962349)

[Incoming Elastic-search backend network traffic 24](#_Toc1201987608)

[Incoming PostgreSQL backend network traffic 25](#_Toc436217869)

[Troubleshooting guide 25](#_Toc1837687017)

[Restore issues 25](#_Toc480506405)

[Error: Database is being accessed by other users 25](#_Toc1934746086)

[Error: Cached artifact not found in offline mode 26](#_Toc549175437)

[Error: Existing arch does not match the requested one. 26](#_Toc1995087369)

[Other Errors: 26](#_Toc1745166756)

[Appendix 28](#_Toc1522583186)

[What to change in config.toml 28](#_Toc2002284643)

[What to write in cidr block 28](#_Toc790914871)

[How to set vpc 28](#_Toc1363312911)

# 

# HA cluster introduction

**High availability (HA)** refers to a system or application (such as a network, a server array, or cluster) that offers a high level of operational performance and quality over a relevant time with maximum potential uptime and accessibility for the content stored on it.

While a more basic system will be adequate to serve content to a low or medium number of users, it may include a single point of failure. This means that if one server goes down, whether due to traffic overload or any number of other issues, the entire site or application could become unavailable.

HA simply means the application remains available with no interruption. We achieve high availability when an application continues to operate when one or more underlying components fail. For example, a router, switch, firewall, or server that fails.

Thus, HA is designed to avoid loss of service by reducing or managing failures and minimizing unscheduled downtime (when your system or network is not available for use or is unresponsive) that happens due to power outages or failure of a component.

“Availability” includes two periods of time: how much time a service is accessible and how much time the system needs to respond to user requests. When it comes to measuring availability, several factors are salient. These include recovery time and both scheduled and unscheduled maintenance periods. Typically, availability is expressed as a percentage of uptime defined by service level agreements (SLAs). A score of 100 percent characterizes a system that never fails or experiences zero downtime by being 100% operational.

# HA Components

This section lists the various Chef Automate High Availability (HA) components and their purpose.

**Automate-ha-ctl**

Aids connect the backend (postgres and elasticsearch) databases using an automate configuration file and Terraform without any manual intervention.

**Automate-ha-curator**

Elasticsearch curator aids in curating and managing the Elasticsearch indices and snapshots by obtaining the entire actionable list of indices (or snapshots) from the cluster. This component is the same as the default curator. It’s written in a HAB package to merge applications in a HAB environment.

**Automate-ha-deployment**

Aids in setting up a workspace for Chef Automate HA environment. For example, /hab/a2\_deploy\_workspace. It also includes terraform code, some necessary scripts, inspecs, tests, Makefile and so on.

**Automate-ha-elasticsearch**

Includes the Elasticsearch configuration and builds the Elasticsearch package. It is installed in the backend nodes.

**Automate-ha-elasticsidecar**

Provides a sidecar service for automate-backend-elasticsearch that reads user’s credentials and passwords of the Elasticsearch binding and applies it to Elasticsearch using the odfe tooling.

**Automate-ha-haproxy**

Aids in sending a request to the leader node and is placed on postgres cluster.

**Automate-ha-pgleaderchk**

This component is used in a proxy health check to determine where to route SQL requests. A golang service that checks the local PostgreSQL instance to view if it’s a Leader.

**Automate-ha-postgresql**

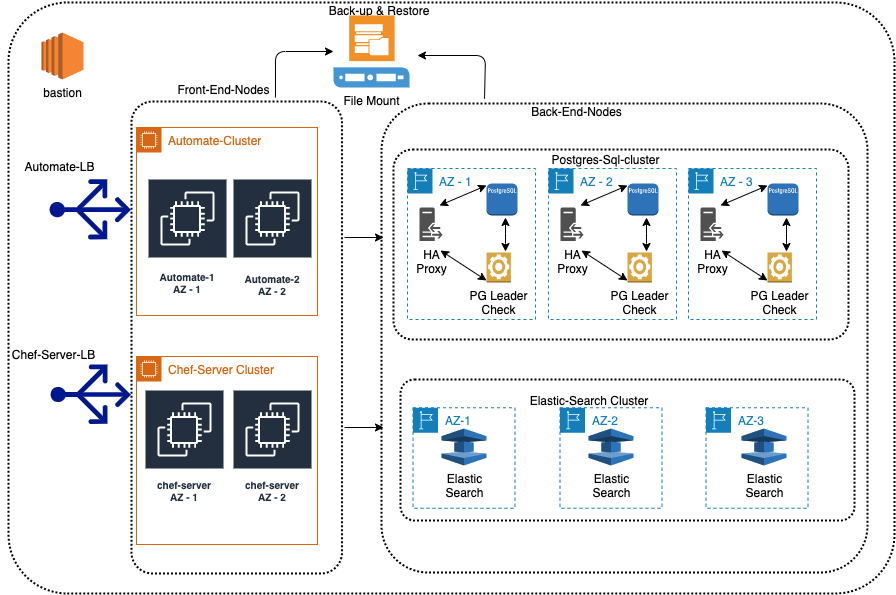
This component is a wrapper package of core/postgresql11 for Chef Automate that provides a backend HA PostgreSQL.

# High availability Architecture

## Reference Architecture

This section includes Chef Automate High Availability (HA) high-level reference architecture that interacts with the HA backend components on different providers or in different environments.

The following Chef Automate HA architecture diagram shows the components involved in the Chef Automate HA that works on Leader-Follower strategy. We are creating the cluster of the Chef Automate, Chef Server, Postgres, and Elasticsearch for Chef Automate HA.



The Chef Automate HA architecture involves two different clusters part of the main cluster, which are:

## Automate Backend Cluster

The backend components connect into the frontend habitat supervisor cluster. In the habitat supervisor, postgres and Elasticsearch instances runs. A minimum of three nodes is required for Postgres and Elasticsearch databases, where one becomes a leader, and others are followers.

## Automate Frontend Cluster

Chef Automate and Chef Server act as frontend nodes and serve as a web UI with load balancer configurations.

These clusters comprise four different servers with HA mode, which are as follows:

Chef-automate

Chef Infra Server

Elasticsearch - an open-source search and analytics engine based on Apache Lucene and built with Java. It is a NoSQL database that stores data in an unstructured way.

PostgreSQL - an open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance.

Elastic Search internally manages the communication and backup and does not follow any leader-follower strategy.

## Deployment Support Types

Currently, Chef Automate HA supports two types of deployment, which are

[Cloud deployment - AWS](#_Configuration_and_Provisioning)

[Bare Infrastructure Deployment](#_Configuration_On-prem)

### AWS Deployment

In [AWS deployment](#_Configuration_and_Provisioning), the entire Chef Automate HA infrastructure is built into the AWS cloud. If you choose AWS as a reference architecture, a standard Terraform script handles AWS deployment. This deployment terraform script first sets up all the prerequisites like creating a VPC, EC2, load balancer, security groups, subnets. Then, ensures the VPCID is provided for security purposes, and the cidr block is created manually based on respective VPC.

Later, following series of configurations and installation is performed:

Installing and configuring PostgreSQL into the postgres instances

Configuring and installing Elasticsearch into Elasticsearch instances

Installing a Chef Habitat and creation of a supervisor network.

Installing automate into the automate instances

Installing Chef Infra Server in all chef-server instances

### Bare Infrastructure Deployment

Bare infrastructure deployment is for those customers who already have basic network infrastructure with VMs, networks, load balancers in their environment. This environment can be on-premises or in the cloud, and the respective organizations might not want to provide access to create items like VMs. In such cases, IPs of their instances are used to set up Chef Automate HA on their network premises.

If you don’t let Terraform create them, or the customer has already made those by themselves, or customers have on-premises servers, or the customers just want to configure Chef Automate HA (automate, chef-server, Elasticsearch, PostgreSQL) in those servers, and then the customer should choose existing\_node reference architecture.

You can also utilize Terraform script for this scenario; however, then this script only handles installing and configuring components and does not create instances on the cloud providers.

# System and software requirements

This section lists the recommended operating systems requirements, virtual machine instances requirements for implementing Chef Automate High Availability (HA) for your network infrastructure or systems or applications or services.

## Platform support

|  |  |
| --- | --- |
| Operating Systems | Tested |
| Red Hat Enterprise Linux (64 Bit OS) | 7, 8 (For 8 or above versions, SELinux configuration must be permissive. By default, in RHEL 8 SELinux configuration is enforced). Red Hat Enterprise Linux derivatives include Amazon Linux v1 (using RHEL 6 |
| Ubuntu (64 Bit OS) | 16.04.x, 18.04.x |
| Centos (64 Bit OS) | 7 |

## Virtual Machine (VM) Instances Type

Based on number of nodes

|  |  |  |
| --- | --- | --- |
| Instance | RAM | Volume-size |
| PostgreSQL | 4 GB RAM for test | 50 GB (dedicated hard disk space assigned to ‘/'). |
| Elasticsearch | 8 GB RAM for test | 50 GB (dedicated hard disk space assigned to ‘/'). |
| Chef Automate | 4 GB RAM for test | 50 GB (dedicated hard disk space assigned to ‘/'). |
| Chef Infra Server | 4 GB RAM for test | 50 GB (dedicated hard disk space assigned to ‘/'). |

ES volume size also depends on the number of nodes and frequency of Chef Infra Client runs and compliance scans. For all the above instances’ RAM and volume size will only for test setup. For production it will depend on number of nodes and frequency of Chef Infra Client runs and compliance scans.

**For Elasticsearch and PostgreSQL, a minimum of three node clusters is required.**

# Bastion host

## Bastion Introduction

A [Bastion Host](https://en.wikipedia.org/wiki/Bastion_host#:~:text=A%20bastion%20host%20is%20a,the%20threat%20to%20the%20computer.) is a special-purpose computer or server on a network specifically designed and configured to withstand attacks. This serve type generally hosts a single application or process, for example, a proxy server or load balancer, and all other services are limited to reduce the threat to the computer.

Its purpose is to provide access to a private network from an external network, such as the Internet or outside of a firewall and involves access from untrusted networks or computers. These computers are also equipped with special networking interfaces to withstand high-bandwidth attacks through the internet.

## Bastion Host Setup

Bastion servers are instances that resides within your public subnet and accessed using SSH. The purpose of a bastion host is to restrict access to a private network from an external network. Once remote connectivity establishes with the bastion host, it allows you to use SSH to login to other instances (within private subnets) deeper within your network.

The bastion hosts provide secure access to Linux instances located in the private and public subnets.

# Getting started

## Package download

Chef-automate is the main utility used for installation of chef-automate. If you are doing installation on fresh server where you don’t have chef-automate utility, you can download it using below link

*curl* [*https://packages.chef.io/files/current/latest/chef-automate-cli/chef-*](https://packages.chef.io/files/current/latest/chef-automate-cli/chef-automate_linux_amd64.zip) *automate\_linux\_amd64.zip | gunzip - > chef-automate && chmod +x chef-automate | cp chef-automate /usr/bin/chef-automate*

# Configuration and Provisioning – Cloud

This section is only for cloud deployment. Currently we support AWS based provisioning and deployment.

## Cloud System Requirements

Please refer to [Common System Requirements](#_System_and_software) for general requirement guidelines.

### Virtual Machine (VM) Instances Type

Based on number of nodes

|  |  |  |  |
| --- | --- | --- | --- |
| Instance | Type | RAM | Volume-size |
| PostgreSQL | t3.medium | 4 GB RAM for test and 8 GB for production. vCPU - 2. | 50 GB (dedicated hard disk space assigned to ‘/'). |
| Elasticsearch | m5.large | 8 GB RAM for test and 16 GB for production. vCPU - 2. | 50 GB (dedicated hard disk space assigned to ‘/'). |
| Chef Automate | t3.medium | 4 GB RAM for test and 8 GB for production. vCPU - 2. | 50 GB (dedicated hard disk space assigned to ‘/'). |
| Chef Infra Server | t3.medium | 4 GB RAM for test and 8 GB for production. vCPU - 2. | 50 GB (dedicated hard disk space assigned to ‘/'). |

ES volume size also depends on the number of nodes and frequency of Chef Infra Client runs and compliance scans. The above table includes AWS instance types. However, for Bare-infra deployment or In-premises deployment types, you can choose the above requirements for VM like RAM.

For Elasticsearch and PostgreSQL, a minimum of three node clusters is required.

### Amazon’s Virtual Private Cloud (VPC)

#### VPC requirements

Amazon VPC, a virtual network dedicated to your AWS account that enables you to launch AWS resources into a virtual network. This virtual network resembles a traditional network that you had operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Amazon VPC is the networking layer for Amazon EC2. Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. Amazon EC2 enables you to scale up or down to handle changes in requirements or spikes in popularity, reducing your need to forecast traffic.

VPC creates an isolated virtual network environment in the AWS cloud, dedicated to your AWS account. Other AWS resources and services operate inside of VPC networks to provide cloud services. AWS VPC looks familiar to anyone used to running a physical Data Center (DC). A VPC behaves like a traditional TCP/IP network that can be expanded and scaled as needed. However, the DC components you are used to dealing with—such as routers, switches, VLANS, etc.—do not explicitly exist in a VPC. They have been abstracted and re-engineered into cloud software.

All VPCs are created and exist in one—and only one—AWS region. AWS regions are geographic locations around the world where Amazon clusters its cloud data centers.

The advantage of regionalization is that a regional VPC provides network services originating from that geographical area. If you need to provide closer access for customers in another region, you can set up another VPC in that region.

This aligns nicely with the theory of AWS cloud computing where IT applications and resources are delivered through the internet on-demand and with pay-as-you-go pricing. Limiting VPC configurations to specific regions allows you to selectively provide network services where they are needed, as they are needed.

Each Amazon account can host multiple VPCs. Because VPCs are isolated from each other, you can duplicate private subnets among VPCs the same way you could use the same subnet in two different physical data centers. You can also add public IP addresses that can be used to reach VPC-launched instances from the internet.

You can modify or use that VPC for your cloud configurations or you can build a new VPC and supporting services from scratch.

#### Amazon’s Virtual Private Cloud (VPC) Limit

Note: - You require a minimum of three node clusters for Elasticsearch and Postgres-sql instances.

AWS limits the size of each VPC; a user cannot change the size once the VPC has been created. Amazon VPC also sets a limit of 200 subnets per VPC, each of which can support a minimum of 14 IP addresses. AWS places further limitations per account / per region, including limiting the number of VPCs to five, the number of Elastic IP addresses to five, the number of Internet gateways per VPC to one, the number of virtual private gateways to five and the number of customer gateways to 50.

CIDR block -Classless Inter-Domain Routing. An internet protocol address allocation and route aggregation methodology. For more information, see Classless Inter-Domain Routing in Wikipedia. Subnet - A range of IP addresses in your VPC.

VPC IP address ranges are defined using Classless interdomain routing (CIDR) IPv4 and IPv6 blocks. You can add primary and secondary CIDR blocks to your VPC, if the secondary CIDR block comes from the same address range as the primary block.

## Configuration

Create a config for aws using below command

./chef-automate init-config-ha aws

## Provisioning

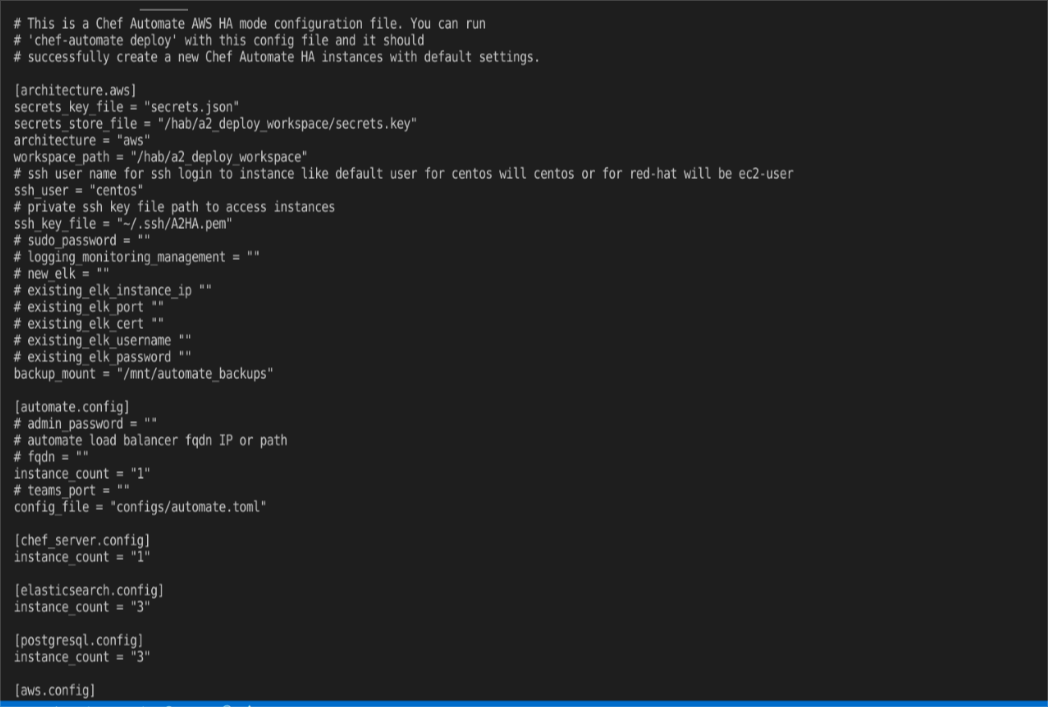
This step is only for cloud deployment. Using provisioning command, we provision the cloud infrastructure as per configuration provided for Automate HA.

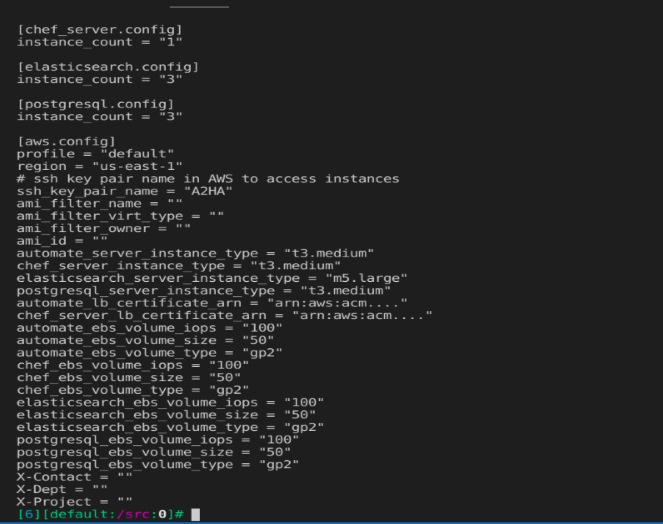
### AWS provisioning

**Setup configuration file for HA Deployment on AWS**  
 */chef-automate provision-infra config.toml*

This will create configuration for deployment on AWS. config.toml is the config file where you need to make changes for any change in Automate HA

By default, config file will look like below:





So here you should make all the changes required for AWS deployment. Refer this doc for config.toml [What to change in config.toml](#_What_to_change)

*./chef-automate provision-infra <path to config.toml>*  
 This step will download habitat and create workspace /hab/a2\_deploy\_workspace for you and this will provision infrastructure for you on AWS

* 1. 

# On-prem Configuration

This section is for configuration related information for on-prem deployment. For this type of deployment user will their own provisioned VMs based on the [System requirements](#_System_and_software). Hence no provisioning step is required for on-prem deployment.

[Bastion host setup](#_Bastion_host)

## On-prem Prerequisite

List of VM with public and private IP. Public-ip is only mandatory for Elasticsearch.

 All the VM must expose the port 22 for SSH.

 We need to open certain port across the VM to make the communication. Please refer this doc for

[Firewall and security settings](#_Security_and_firewall) that need to be done before deployment.

|  |  |
| --- | --- |
| **Component** | **Port** |
| Habitat gossip (UDP) | **9638** |
| Habitat http API | **9631** |
| PostgreSQL | **5432** |
| Pgleaderchk | **6432** |
| HaProxy | **7432** |
| Elasticsearch (https) | **9200** |
| Elasticsearch (transport) | **9300** |
| Kibana | **5601** |

## Configuration

Setup configuration file for HA Deployment on AWs   
*./chef-automate init-config-ha existing\_infra*

This will create configuration for deployment on existing nodes. Config.toml is the config file where you need to make changes for any change in Automate HA   
By default, config file will look like below:Text

Description automatically generated

At the end in existing\_infra config part, you need to provide IP’s of your on premise details separated by comma. we must mention the List of IP address for the cluster. In the below image there are options for private and public ip's (public IP is needed for elastic search only).

Text

Description automatically generated 

# Validation

Validation command work is in progress. This command will internally call a set of scripts and trigger the checks for required firewalls and security settings. This is a pre-deployment step.

# Installation

*./chef automate deploy  <path to config.toml>*   
This will generate workspace and download the habitat on your system. 

./*chef-automate info*

This will give information about all server’s IP and automate’s URL details.

*./chef-automate status*

This will give the status of frontend and backend node.

## Air-gapped installation

Need to ad steps for air-gapped installation.

# Backup and restore

Back-up configurations to be done before deploying cluster.

## Pre-back-up configuration:

### ES configuration and setup

A shared file system is needed to create Elasticsearch snapshots. In order to register the snapshot repository with Elasticsearch it is necessary to mount the same shared filesystem to the same location on all master and data nodes. This location (or one of its parent directories) must be registered in the path.repo setting on all master and data nodes.

Assuming that the shared filesystem is mounted to /mnt/automate\_backups, we can configure Automate to register the snapshot locations with Elasticsearch.

Ensure the shared file system is mounted on all Elasticsearch servers:

*mount /mnt/automate\_backups*

Create elasticsearch sub-directory and set permissions, this will only need to be done on a single Elasticsearch server if the network mount is correctly mounted.

*sudo mkdir /mnt/automate\_backups/elasticsearch*

*sudo chown hab:hab /mnt/automate\_backups/elasticsearch/*

Configure Elasticsearch path.repo setting by SSHing to a single Elasticsearch server and using the following steps:

Export the current Elasticsearch config from the Habitat supervisor. You will need to have root access to run the following commmands

*source /hab/sup/default/SystemdEnvironmentFile.sh*

*automate-backend-ctl applied --svc=automate-ha-elasticsearch | tail -n +2 > es\_config.toml*

Edit es\_config.toml and add the following settings to the end of the file.  
Note: If credentials have never been rotated this file may be empty.

***[es\_yaml.path]***

***repo = "/mnt/automate\_backups/elasticsearch"***

Apply updated es\_config.toml config to Elasticsearch, this only needs to be done once. This will trigger a restart of the Elasticsearch services on each server.

*hab config apply automate-ha-elasticsearch.default $(date '+%s') es\_config.toml*

hab svc status (check elasticsearch service is up or not)

curl -k -X GET "[https://localhost:9200/\_cat/indices/\*?v=true&s=index&pretty](https://localhost:9200/_cat/indices/*?v=true&s=index&pretty)" -u admin:admin (Another way to check es. Check that all the indices is green or not)

Configure Automate to handle external Elasticsearch backups

Create a automate.toml file on the provisioning server

touch automate.toml

Add the following configuration to automate.toml on the provisioning host.

[global.v1.external.elasticsearch.backup]

enable = true

location = "fs"

[global.v1.external.elasticsearch.backup.fs]

# The `path.repo` setting you've configured on your Elasticsearch nodes must be a parent directory of the setting you configure here:

path = "/mnt/automate\_backups/elasticsearch"

[global.v1.backups.filesystem]

path = "/mnt/automate\_backups/backups"

After that patch the config. This will trigger also the deployment.

./chef-automate config patch automate.toml

### S3 Configuration for backup

In order to run the terraform scripts, we need an IAM user with proper permissions. All the required permissions are mentioned in the next section. We need to make sure that we have the access key id and secret access key for the user. If not, then regenerate a new access key and keep it handy.

Permissions to be provided:

We need to check if the IAM user has all the required permissions or not. Listed below are the must have permission policies:

AdministratorAccess

AmazonAPIGatewayAdministrator

AmazonS3FullAccess

We also have to create IAM role to give access of s3 to elasticsearch iinstances.

These permissions can either be directly added to the user or can be added via IAM Group.

After doing the above steps, we need to create toml file and patch the config. Please modify the below listed values in the file:

bucket name (bucket = "bucket-name" and name = "bucket-name")

mkdir configs

vi configs/automate.toml

**Put below content in automate.toml file:**

[global.v1.external.elasticsearch.backup]

enable = true

location = "s3"

[global.v1.external.elasticsearch.backup.s3]

# bucket (required): The name of the bucket

bucket = "bucket-name"

# base\_path (optional): The path within the bucket where backups should be stored

# If base\_path is not set, backups will be stored at the root of the bucket.

base\_path = "elasticsearch"

# name of an s3 client configuration you create in your elasticsearch.yml

# see <https://www.elastic.co/guide/en/elasticsearch/plugins/current/repository-s3-client.html>

# for full documentation on how to configure client settings on your

# Elasticsearch nodes

client = "default"

[global.v1.external.elasticsearch.backup.s3.settings]

## The meaning of these settings is documented in the S3 Repository Plugin

## documentation. See the following links:

## <https://www.elastic.co/guide/en/elasticsearch/plugins/current/repository-s3-repository.html>

## Backup repo settings

# compress = false

# server\_side\_encryption = false

# buffer\_size = "100mb"

# canned\_acl = "private"

# storage\_class = "standard"

## Snapshot settings

# max\_snapshot\_bytes\_per\_sec = "40mb"

# max\_restore\_bytes\_per\_sec = "40mb"

# chunk\_size = "null"

## S3 client settings

# read\_timeout = "50s"

# max\_retries = 3

# use\_throttle\_retries = true

# protocol = "https"

[global.v1.backups]

location = "s3"

[global.v1.backups.s3.bucket]

# name (required): The name of the bucket

name = "bucket-name"

# endpoint (required): The endpoint for the region the bucket lives in.

# See <https://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region>

endpoint = "[https://s3.amazonaws.com](https://s3.amazonaws.com/)"

# base\_path (optional): The path within the bucket where backups should be stored

# If base\_path is not set, backups will be stored at the root of the bucket.

base\_path = "automate"

[global.v1.backups.s3.credentials]

# Optionally, AWS credentials may be provided. If these are not provided, IAM instance

# credentials will be used. It's also possible for these to be read through the standard

# AWS environment variables or through the shared AWS config files.

# Use the credentials obtained from here [AWS-Credential](https://github.com/chef/automate-as-saas/wiki/Bastion-Setup#aws-credentials)

access\_key = "AKIARUQHMSKHGYTUJ&UI”

secret\_key = "s3kQ4Idyf9WjAgRXyv9tLYCQgYTRESDFRFV"

After putting contents in automate.toml file, we need to eceute below command. This command will also trigger the deployment.

**./chef-automate patch configs/automate.toml**

Back-up configurations to be done after deploying cluster

IAM Role: Assign the IAM Role to the all the elastic search instances in the cluster that we create above step.

### File System (EFS)Configuration for backup

Backup on share file system. (This section is specific for aws).

Create the EFS over the AWS.

Once EFS is ready there are 2 ways to mount (via DNS and via IP).

Open the port(2049) Proto(NFS) for EFS security group.

## Backup

To create a new backup run chef-automate backup create from a Chef-Automate front-end node.

*./chef-automate backup create*

## Restore

Check status of all Chef Automate and Chef Infra Server front-end nodes.

*chef-automate status*

Shutdown Chef Automate service on all front-end nodes.

*sudo systemctl stop chef-automate*

Login to same instance of Chef Automate front-end node from which backup is taken run the restore command

chef-automate backup restore --yes -b /mnt/automate\_backups/backups --patch-config /etc/chef-automate/config.toml

Start all Chef Automate and Chef Infra Server front-end nodes.

sudo systemctl start chef-automate

**In case of S3 back-up:**

Login to same instance of Chef Automate front-end node from which backup is taken run the restore command

chef-automate backup restore s3://bucket\_name/path/to/backups/BACKUP\_ID --skip-preflight --s3-access-key "Access\_Key" --s3-secret-key "Secret\_Key"

Start all Chef Automate and Chef Infra Server front-end nodes.

sudo systemctl start chef-automate

# Upgrade

# Migration

## Chef server (HA- backend) to Automate HA

If existing customer wants to move its existing chef infrastructure to our new a2-ha-backend cluster, it needs to do migration.

**For that we have identified there can be 2 scenarios**

1. Migrating from standalone chef-server to automate chef-server which is part of a2-ha-backend frontend nodes cluster

2. Migrating from chef-backend cluster to automate chef-server which is part of a2-ha-backend frontend nodes cluster

In both the cases we need to take backup using knife-ec-backup utility and then move the backup folder on the new chef-server where will take restore using the same utility. This backup will migrate all the cookbooks, users, data-bags, policies and organisations.

knife-ec-backup can backup and restore the data in an Enterprise Chef Server installation, preserving the data in an intermediate, editable text format. It is similar to the knife download and knife upload commands and uses the same underlying libraries, but also includes workarounds for objects not yet supported by those tools and various Server API deficiencies. The long-run goal is to improve knife download, knife upload and the Chef Infra Server API and deprecate this tool.

**This plugin currently supports Enterprise Chef 11 and Chef Infra Server 12+ .**

### Backup on your existing chef-server:

Install habitat

`curl <https://raw.githubusercontent.com/habitat-sh/habitat/master/components/hab/install.sh> | sudo bash`

Install the habitat package for knife-ec-backup

`hab pkg install chef/knife-ec-backup`

Generate a knife tidy server report to examine stale nodes and unused cookbooks

`hab pkg exec chef/knife-ec-backup knife tidy server report --node-threshold 60`

--node-threshold NUM\_DAYS Maximum number of days since last checking before node is considered stale

Initiate a backup of your Chef Server data

`hab pkg exec chef/knife-ec-backup knife ec backup -c /etc/opscode/chef-server.rb backup\_$(date '+%Y%m%d%H%M%s') --webui-key /etc/opscode/webui\_priv.pem --with-user-sql --with-key-sql`

`--with-user-sql` This is required to correctly handle user passwords and to ensure user-specific association groups are not duplicated.

`--with-key-sql` is to handle cases where customers have users with multiple pem keys associated with their user or clients. The current chef-server API only dumps the default key and sometimes users will generate and assigned additional keys to give additional users access to an account but still be able to lock them out later without removing everyones access.

Run knife tidy server cleans to delete unused data from the reports above.

`hab pkg exec chef/knife-ec-backup knife tidy server clean --backup-path /path/to/an-ec-backup`

Copy backup

Copy the ec backup directory to any Chef Server frontend for the restore target Automate Cluster HA using rsync, NFS, etc or simply copy the folder to chef-server

`scp -i /path/to/key backup\_$(date '+%Y%m%d%H%M%s') user@host:/home/user`

### Restore to Chef Automate HA chef-server

Install the habitat package for knife-ec-backup

`hab pkg install chef/knife-ec-backup`

Restore the backup

`hab pkg exec chef/knife-ec-backup knife ec restore /home/centos/backup\_2021061013191623331154 -yes --concurrency 1 --webui-key /hab/svc/automate-cs-oc-erchef/data/webui\_priv.pem --purge -c /hab/pkgs/chef/chef-server-ctl/14.1.0/20210225010004/omnibus-ctl/spec/fixtures/pivotal.rb`

## Existing A2HA to Automate HA

In some scenario we are required to migrate A2HA data to Automate HA cluster (as we have new HA implementation in Automate). For that here are some steps that you'll have to follow.

Take backup of chef-automate from A2HA (Old) using below command, this command can be executed from any of front-end (chef-automate) node  
in case of multiple frontends. Usually if you don't specify any location for backup in config.toml then that backup will be store on /var/opt/chef-automate/backup location if you hit below command.

sudo chef-automate backup create

Make a tar file of backup that we have taken in step 1. Here make sure that you are also taking backup of .tar directory. Otherwise you'll face some issue related to metadata.json.

E.g. tar -cvf backup.tar.gz path/to/backup/20201030172917/ /path/to/backup/automatebackup-elasticsearch/ /path/to/backup/.tmp/

Create a aib file from any of chef-automate frontend node of A2HA. This will create a bundle of all necessary keys. Like pivotal.pem, secret key etc. Usually this will not be included in regular backup(step 1) so make sure you create a bundle for that.

sudo chef-automate bootstrap bundle create bootstrap.abb

Copy tar file and aib file that we created in step 2 and 3 respectively to any of the Automate HA chef-automate instance and extract it on specific location that you mentioned in config.toml file. Its is not necessary to extract that backup on below location. But make sure that restore command is able to read backup or not from your defined location on step1.

E.g /mnt/automate-backup

Restore A2HA backup on Automate HA. Read this docs for [chef-automate restore](https://docs.chef.io/automate/restore/). In below command there is also a frontend-20210624095659.aib generated in new Automate HA file mention that's because while restoration we also keep in mind all the services habitat release version. Because during restoration time A2HA restoration will try to find A2HA habitat pkg version so there can be a scenario occure where all(A2HA and automate HA frontends (automate's)) packages version can't be the same. That's why we are passing current Automate HA packages. You can find frontend aib file in /var/tmp directory of your Automate HA chef-automate instance.

E.g. sudo chef-automate backup restore /mnt/automate\_backups/backups/20210622065515/ --patch-config /etc/chef-automate/config.toml --airgap-bundle /var/tmp/frontend-20210624095659.aib --skip-preflight

After that if you not do this step then you will might face warning when you'll try to open chef-automate UI It looks like you do not have permission to access any data in automate So make sure you have unpacked the. aib file. Otherwise you'll not see login page. To unpack the bootstrap file that we copied from A2HA chef-automate using below command

Sudo chef-automate bootstrap bundle unpack bootstrap.abb

Copy the bootstrap.aib file to another automate node and chef node also if you are having multiple automate and chef instances. Because the secrets we have restored by unpacking the bootstrap file would be different for another automate instance. So we need to make that all the automate and chef instance would be in sync.

Important command and notes

Using the below command you can see what bootstrap includes in aib file and abb file. Aib file will only include keys related data while abb file will include service packages also. You can use below command and can compare both the file's data

`tail -n +3 bootstrap.aib | tar -tf -`

After using above command, if you want to see the data of service like secret service then you would see those services into /hab/svc directory. This'll be needed if you want to compare aib data between multiple FE(In respective chef-automate and chef-server) nodes.

E.g For secret service `cat /hab/svc/secrets-service/data/secrets\_key`

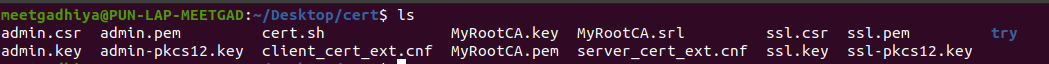
## A2 to Automate HA

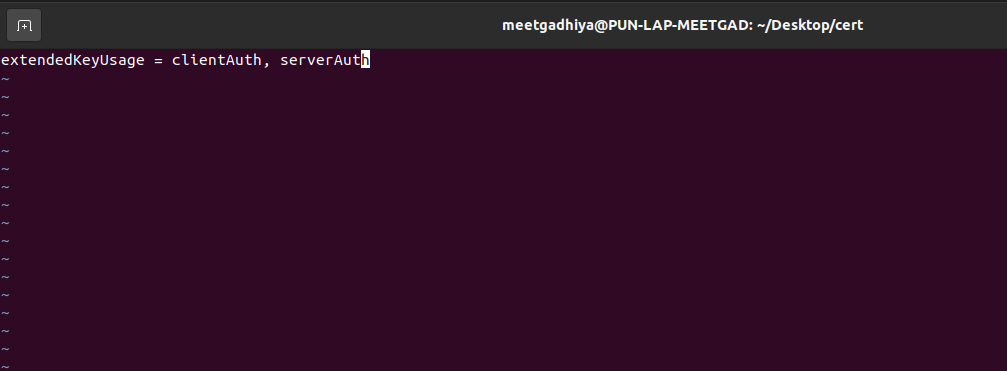
# Performance benchmarking

# Certificates renewal

Create "server\_cert\_ext.cnf" and "client\_cert\_ext.cnf" file and paste below content in both the file. Both files should be in same dir where openssl script is placed. As we see in below image cert.sh.

extendedKeyUsage = clientAuth, serverAuth





Use the script below to generate certs. This is a bash script content so copy this content to a new file and execute it from bastion and this script will put certs at /hab/a2\_deploy\_worspace/certs directory. This will need a openssl utility so make sure that this utility is installed.

----------------------------------------------------------cert.sh------------------------------------------------------------------

#!/bin/bash

openssl genrsa -out ca\_root.key 2048

openssl req -x509 -new -key ca\_root.key -sha256 -out ca\_root.pem -subj '/C=US/ST=Washington/L=Seattle/O=Chef Software Inc/CN=chefrootca'

openssl genrsa -out admin-pkcs12.key 2048

openssl pkcs8 -v1 "PBE-SHA1-3DES" -in "admin-pkcs12.key" -topk8 -out "es\_admin\_ssl\_private.key" -nocrypt

openssl req -new -key es\_admin\_ssl\_private.key -out admin.csr -subj '/C=US/ST=Washington/L=Seattle/O=Chef Software Inc/CN=chefadmin'

openssl x509 -req -in admin.csr -CA ca\_root.pem -CAkey ca\_root.key -CAcreateserial -out es\_admin\_ssl\_public.pem -sha256 -extfile server\_cert\_ext.cnf

openssl genrsa -out ssl-pkcs12.key 2048

openssl pkcs8 -v1 "PBE-SHA1-3DES" -in "ssl-pkcs12.key" -topk8 -out “es\_ssl\_private.key” -nocrypt

openssl req -new -key es\_ssl\_private.key -out ssl.csr -subj '/C=US/ST=Washington/L=Seattle/O=Chef Software Inc/CN=chefnode'

openssl x509 -req -in ssl.csr -CA ca\_root.pem -CAkey ca\_root.key -CAcreateserial -out es\_ssl\_public.pem -sha256 -extfile client\_cert\_ext.cnf

cp ca\_root.pem /hab/a2\_deploy\_workspace/certs/ca\_root.pem

cp es\_admin\_ssl\_public.pem /hab/a2\_deploy\_workspace/certs/es\_admin\_ssl\_public.pem

cp es\_admin\_ssl\_private.key /hab/a2\_deploy\_workspace/certs/es\_admin\_ssl\_private.key

cp es\_ssl\_public.pem /hab/a2\_deploy\_workspace/certs/es\_ssl\_public.pem

cp es\_ssl\_private.key /hab/a2\_deploy\_workspace/certs/es\_ssl\_private.key

cp es\_admin\_ssl\_private.key /hab/a2\_deploy\_workspace/certs/kibana\_ssl\_private.key

cp es\_admin\_ssl\_public.pem /hab/a2\_deploy\_workspace/certs/kibana\_ssl\_public.pem

cp es\_ssl\_private.key /hab/a2\_deploy\_workspace/certs/pg\_ssl\_private.key

cp es\_ssl\_public.pem /hab/a2\_deploy\_workspace/certs/pg\_ssl\_public.pem

-------------------------------------------------------------------------------------------------------------------------------------

Now copy the content of the certs in an appropriate file. See below steps

Execute this command from /hab/a2\_deploy\_workspace. This command will craete skeleton of certificate.

*./scripts/credentials set ssl --rotate-all*

Execute bash script that we created above

bash cert.sh

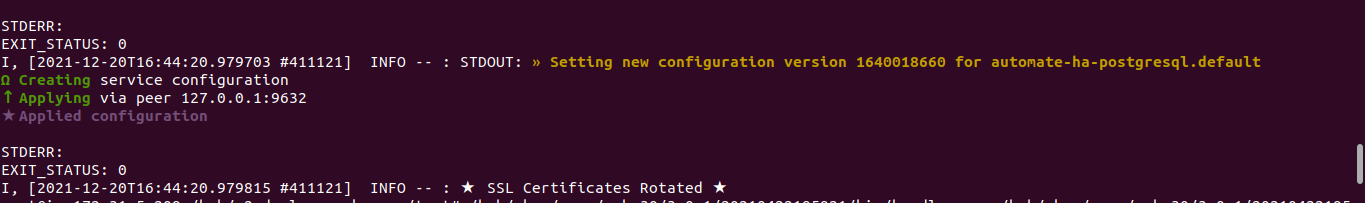
Now apply first es ssl from /hab/a2\_deploy\_workspcae

*./scripts/credentials set ssl --es-ssl*

*./scripts/credentials set ssl --pg-ssl*

*./scripts/credentials set ssl --kibana-ssl*

After successfully apply the certificates, you will see the the output like this.



Go to automate and chef server instance and check the chef service status. If you see the service down or critical, then just wait for 3-4 min because after applying the certs it will take around 3-4 min to up.

# Security and firewall

Automate Cluster requires several ports to be open between the Frontend and Backend servers in order to operate. Below it a breakdown of those ports and what needs to be open to each set of servers.

## Incoming frontends network traffic

Provisioning server => Frontends

TCP 22 - This allows terraform to ssh in and configure services

TCP 9631 - This allows our tools to query information from the backend to configure Automate

Users/chef-servers/chef-clients => Frontends

TCP 443 - This allows users to reach the UI and chef-servers to reach the api for reporting. If chef-clients report directly or download profiles, they'll need 443 access as well

TCP 80 - Optional, however if not in place something should redirect users to 443 before they reach it

## Incoming Elastic-search backend network traffic

Provisioning server => elasticsearch-backends

TCP 22 - This allows terraform to ssh in and configure services

TCP 9631 - This allows our tools to query information from the backend to configure Automate

admin-users => elasticsearch-backends

TCP 5601 - Optional, this allows admins to reach Kibana on the Elasticsearch servers. This hosts an operations dashboard that shows metrics for the Elasticsearch and Postgres servers.

Frontend => elasticsearch-backends

TCP 9200 - This allows Automate to talk to the Elasticsearch API

TCP 9631 - This allows our tools to query information from the backend to configure Automate

elasticsearch-backends <=> elasticsearch-backends

TCP 9300 - This allows Elasticsearch to distribute data in its cluster

TCP/UDP 9638 - This allows Habitat to communicate configuration changes between elasticsearch nodes

TCP 9631 - This allows the Habitat API to be reachable from services on the elasticsearch nodes

postgres-backends <=> elasticsearch-backends

TCP 9200 - This allows metricbeats to send data to elasticsearch for use in Kibana

TCP/UDP 9638 - This allows Habitat to communicate configuration changes between all backend nodes

TCP 9631 - This allows the Habitat API to be reachable from services on the all backend nodes

Incoming PostgreSQL backend network traffic

Provisioning server => postgres-backends

TCP 22 - This allows terraform to ssh in and configure services

TCP 9631 - This allows our tools to reach the habitat API for configuration

Frontend => postgres-backends

TCP 7432 - This allows Automate to connect to haproxy which forwards to the psql leader

TCP 9631 - This allows our tools to query information from the backend to configure Automate

postgres-backends <=> postgres-backends

TCP 5432 - This allows haproxy on postgres-backends to forward connections to the leader

TCP/UDP 9638 - This allows Habitat to communicate configuration changes between postgres nodes

TCP 9631 - This allows the Habitat API to be reachable from services on the postgres nodes

elasticsearch-backends <=> postgres-backends

TCP/UDP 9638 - This allows Habitat to communicate configuration changes between all backend nodes

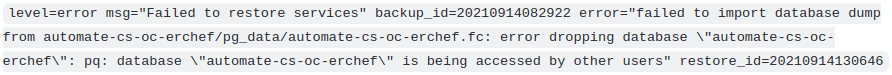
TCP 9631 - This allows the Habitat API to be reachable from services on all backend nodes

# Troubleshooting guide

## Restore issues

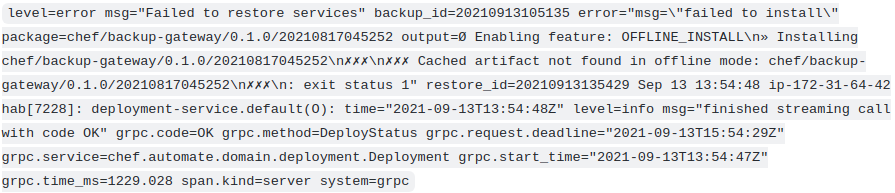
Below are a few frequently encountered issues in Restore and steps on how to resolve them:

### Error: Database is being accessed by other users



**How to resolve:** Some services may be still in running state and are referring to the database, while restore service is trying to drop the database. Please check if all the front end and backend services are stopped.

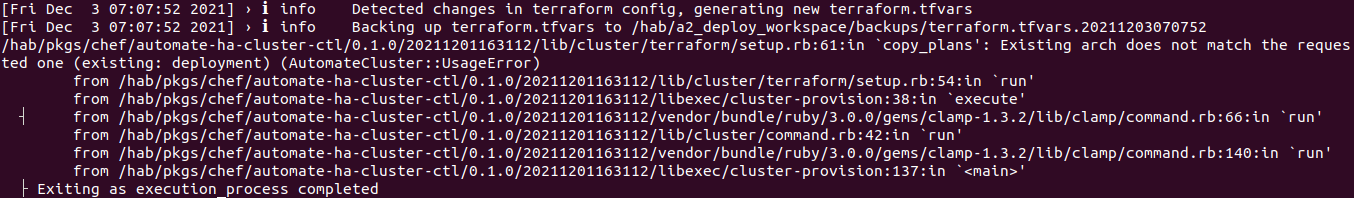
Error: Cached artifact not found in offline mode



How to resolve: In case of this error, we need to use the --airgap-bundle option along with the restore command. Please find the name of the airgap bundle from the path /var/tmp. the airgap bundle file would be something like frontend-20210908093242.aib

Command : chef-automate backup restore s3://bucket\_name/path/to/backups/BACKUP\_ID --patch-config </path/to/patch.toml> --skip-preflight --s3-access-key "Access\_Key" --s3-secret-key "Secret\_Key" --airgap-bundle /var/tmp/<airgap-bundle>

### Error: Existing arch does not match the requested one.



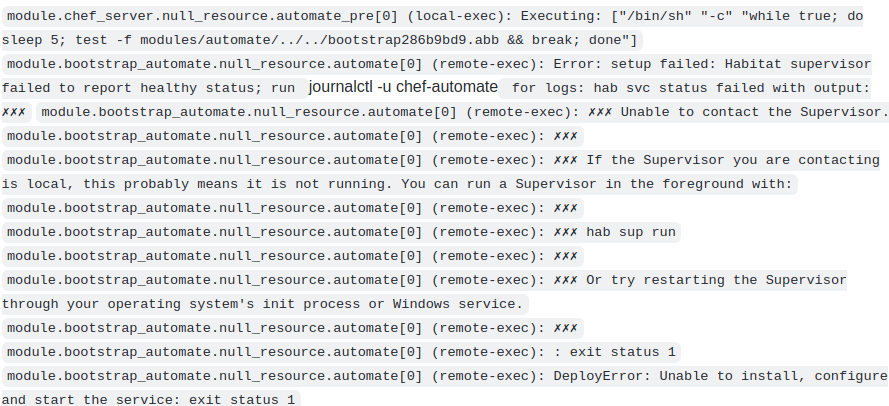
How to Resolve:

Execute belo command from bastion from any location.

sed -i 's/deployment/aws/' /hab/a2\_deploy\_workspace/terraform/.tf\_arch

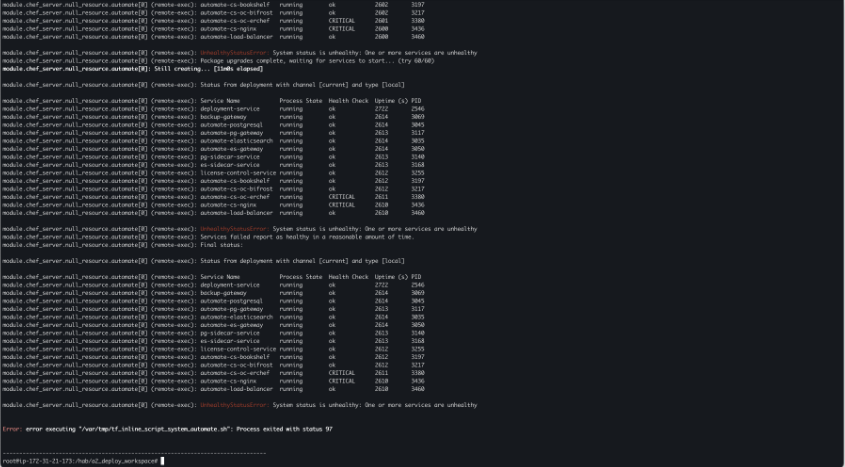
sed -i 's/architecture "deployment"/architecture "aws"/' /hab/a2\_deploy\_workspace/a2ha.rb

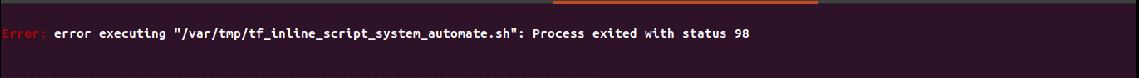
### Other Errors:



After running the following deployment command, the deployment repeatedly fails due to UnhealthyStatusError (refer screenshot)

(./chef-automate deploy config.toml)Those error can occur during deployment time.





In this above all cases, do the things below.

1) ssh into all frontends (automate and chef\_server)

2) From all frontends nodes remove /hab dir and also remove /var/tmp contens.

*rm –rf /hab && cd /var/tmp && rm –rf \**

*sudo kill -9 $(sudo lsof -t -i:9631)*

*sudo kill -9 $(sudo lsof -t -i:9638)*

3) Do terraform destroy of deployment

for i in 1;do i=$PWD;cd /hab/a2\_deploy\_workspace/terraform/;terraform destroy;cd $i;done

After that do deploy again using below command

*./chef-automate deploy config.toml*

# Appendix

## [What to change in config.toml](https://progresssoftware.sharepoint.com/sites/ChefCoreC/_layouts/15/doc.aspx?sourcedoc=%7bac26b0b0-9621-4d83-a6ef-47c363a9aaf7%7d&action=edit)

1. Specify the ssh username and the ssh\_key\_file path. This path should be from bastion and If you scroll down in config.toml then you will find here this key pair name and key file both should have a same content. Suppose you have mentioned the "a2ha-aws" name in key\_pair section then put that file content in ssh\_key\_file path's file.

2. Assign permision for this file "ssh\_key\_file"

chmod 400 /root/.ssh/id

3. Provide the number of node for the respective cluster for chef\_automate and chef\_server otherwise 1 is fine. For PG,ES and chef-server we have to just maintain the cluster number.

Type of instance (Before proceeding ahead, ensure that the instance type mentioned inside a2ha.rb file is supported in the region) volume type, size and iops aws region ssh\_key\_pair\_name (this is same key-pair name what we have used to provision the bastion ec2 machine). In Section-I we have define the ssh\_key\_file, this should point to the same key file as mentioned in 1st step. Setup the secrets management key and any needed passwords. The default location for the secrets key and secret storage is set in the config file. The default location for the key is /etc/chef-automate/secrets.key and the secret store file is in /hab/a2\_deploy\_workspace/secrets.json.

4. You have to provide vpcid and cidr block. a2ha will not create a vpc so you have to provide vpc. You can use the default vpc that you'll find in aws VPC section. And also provide cidr block. Use this doc for cidr block [what to write in cidr block](#_What_to_write)

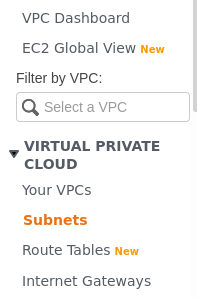
## W[hat to write in cidr block](https://progresssoftware.sharepoint.com/:w:/r/sites/ChefCoreC/_layouts/15/Doc.aspx?sourcedoc=%7B157FBD37-787E-420F-A9F5-9D03A63F2199%7D&file=Vpc%20Instruction.docx&action=default&mobileredirect=true)

How to set vpc  
  
1. Copy VPC ID from aws web portal as shown in the below image in this field in config.toml.

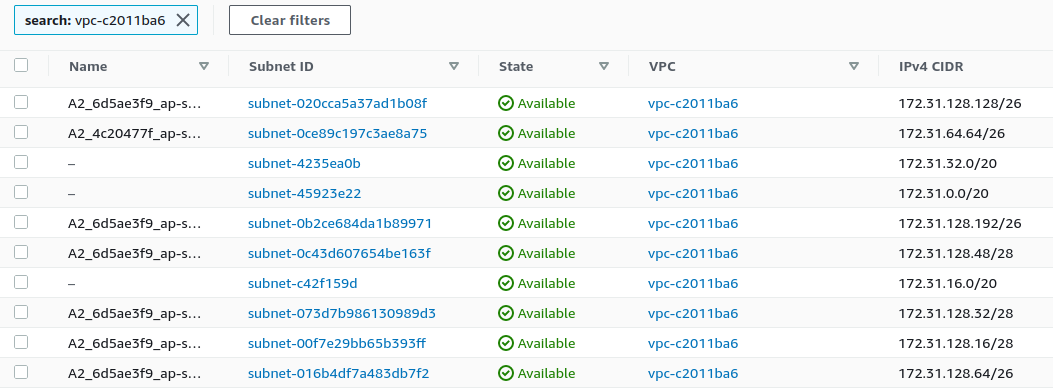
aws\_vpc\_id = “vpc-c2011ba6”



2. For Cidr block first go into the subnet section as shown below image.



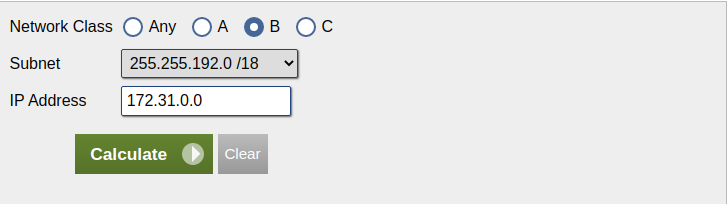
3. Then check into the Ipv4 CIDR field in the subnet section in aws web console.



Now as shown in above image IPv4 CIDR we have to pick a unique one. To choose value for ‘aws\_cidr\_block\_addr’ follow below approach

Use This to Select Correct CIDR Block

<https://www.calculator.net/ip-subnet-calculator.html?cclass=a&csubnet=20&cip=172.31.0.0&ctype=ipv4&printit=0&x=82&y=36>

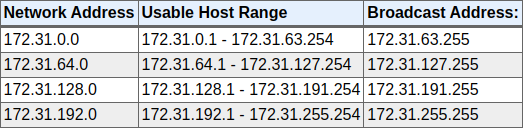


Network class: 172.31.0.0 is a class B ip address so in above image B is selected.

Subnet field: if your vpc IPv4 CIDR block is 172.31.0.0/16 then just add plus 2 to make 18 because we have set /18 in our system. So accordingly, you have to set from subnet drop down. That’s why in above image subnet is selected to /20.

IP Address: Write down 172.31.0.0 from 172.31.0.0/16 from IPv4 CIDR block and press calculate.

After that just scroll up to see below table. In this table you’ll find Network Address field. From that just pick one of them and put into this field‘aws\_cidr\_block\_addr= ‘and do not forgot check that selected address is already occupied or not. Refer step 3 to check.



So as above table give us network address pick unique one. Make sure it should not be conflicted already created subnet. See 3rd step aws subnet image. As seen in that image all addresses have been picked up so we can’t use that vpc id for this deployment. So, we have to change vpc.

However, if you feel subnet is not possible using above calculator then follow ccna docs. Because networking itself a wider topic so not possible to include all the things here.