

Deploying CockroachDB with Terraform Essentials

Table of Contents

1. Introduction to Distributed Databases:3
2. Getting Started with Terraform: 22
3. Initial Deployment of CockroachDB with Terraform: 54
4. Basic Operations in CockroachDB Exploring the CockroachDB interface: 65
5. Introduction to CockroachDB Clusters: 84
6. Expanding Terraform Knowledge: 102
7. Security Fundamentals in CockroachDB :113
8. Backup and Recovery in CockroachDB:130
9. Simple Monitoring and Troubleshooting:172

Introduction to Distributed Databases

What is a distributed database and how do they work?

What is a distributed database?

- A distributed database is a database that runs and stores data across multiple computers, as opposed to doing everything on a single machine.
- Typically, distributed databases operate on two or more interconnected servers on a computer network. Each location where a version of the database is running is often called an instance or a node.
- A distributed database, for example, might have instances running in New York, Ohio, and California. Or it might have instances running on three separate machines in New York.

What is a distributed database used for?

There are different types of distributed databases and different distributed database configuration options, but in general distributed databases offer several advantages over traditional, single-instance databases:

- Distributing the database increases resilience and reduces risk.
- Distributed databases are generally easier to scale.
- Distributing the database can improve performance.
- Geographically distributing the database can reduce latency.

Types of distributed databases: NoSQL vs. distributed SQL databases

- Broadly, there are two types of distributed databases: NoSQL and distributed SQL. (Document-based and key-value are two other terms often used to describe NoSQL databases, so you may sometimes see these options compared as “document based vs. relational,” for example).
- To understand the difference between them, it’s helpful to take a quick dive into the history of databases.
- Humans have been storing data in various formats for millennia, of course, but the modern era of computerized databases really began with Edgar F. Codd and the invention of the relational (SQL) database.
- Relational databases store data in tables and enforce rules – called schema – about what types of data can be stored where, and how the data relate to each other.

Distributed database configurations: active-passive vs. active-active vs. multi-active

- One of the main goals of a distributed database is high availability: making sure the database and all of the data it contains are available at all times.
- But when a database is distributed, its data is replicated across multiple physical instances, and there are several different ways to approach configuring those replicas.

Active-passive

The first, and simplest, is an active-passive configuration. In an active-passive configuration, all traffic is routed to a single “active” replica, and then copied to the other replicas for backup.

- If the data is replicated synchronously (immediately) and writing to one of the “follower” replicas fails, then you must either sacrifice availability (the database will become unavailable unless all three replicas are online) or consistency (the database may have replicas with conflicting data, as an update can be written to the active replica but fail to write to one of the passive follower replicas).

Active-active

- In active-active configurations, there are multiple active replicas, and traffic is routed to all of them. This reduces the potential impact of a replica being offline, since other replicas will handle the traffic automatically.
- However, active-active setups are much more difficult to configure for most workloads, and it is still possible for consistency issues to arise if an outage happens at the wrong time.
- For example, imagine an active-active system with replicas A and B:
- A receives a write for key xyz with the value 123, and then immediately fails and goes offline. A subsequent read for xyz is thus routed to B, and returns NULL, because xyz = 123 hadn't yet been copied to B when A went offline. The application, seeing that there isn't a current value for xyz, sends an xyz = 456 write to B. A comes back online.

Multi-active

- Multi-active is the system for availability used by CockroachDB, which attempts to offer a better alternative to active-passive and active-active configurations.
- Like active-active configurations, all replicas can handle both reads and writes in a multi-active system.
- But unlike active-active, multi-active systems eliminate the possibility of inconsistencies by using a consensus replication system, where writes are only committed when a majority of replicas confirm they've received the write.
- A majority of replicas thus define what is correct, allowing the database to remain both online and consistent even if some replicas are offline at the time of writing.
- If a majority of replicas are offline, the entire database becomes unavailable to prevent the introduction of inconsistent data.

Distributed databases vs. cloud databases

- Since we're discussing configuration options for distributed databases, it's worth pointing out that although the terms distributed database and cloud database are sometimes used interchangeably, they're not necessarily the same thing.
- A distributed database is any database that's distributed across multiple instances.
- Often, these instances are deployed to a public cloud provider such as AWS, GCP, or Azure, but they don't have to be. Distributed databases can also be deployed on-premises, and some even support hybrid cloud and multi-cloud deployments.
- A cloud database is any database that's been deployed in the cloud (generally a public cloud such as AWS, GCP, or Azure), whether it's a traditional single-instance deployment or a distributed deployment.

How a distributed database works

- Distributed databases are quite complicated, and entire books could be written about how they work. That level of detail is outside the scope of this article, but we will take a look at how one distributed SQL database, CockroachDB, works at a high level.
- From the perspective of your application, CockroachDB works very similarly to a single Postgres instance – you connect and send data to it in precisely the same way. But when the data reaches the database, CockroachDB automatically replicates and distributes it across three or more nodes (individual instances of CockroachDB).
- To understand how this occurs, let's focus on what happens to a single range – a chunk of data – when it's written to the database.
- For the purposes of simplicity, we'll use the example of a three-node, single-region cluster, although CockroachDB can support multi-region deployments and many, many more nodes.

Using this guide

- This guide is broken out into pages detailing each layer of CockroachDB.
- We recommended reading through the layers sequentially, starting with this overview and then proceeding to the SQL layer.
- If you're looking for a high-level understanding of CockroachDB, you can read the Overview section of each layer.
- For more technical detail — for example, if you're interested in contributing to the project — you should read the Components sections as well.

Goals of CockroachDB

CockroachDB was designed to meet the following goals:

- Make life easier for humans. This means being low-touch and highly automated for operators and simple to reason about for developers.
- Offer industry-leading consistency, even on massively scaled deployments. This means enabling distributed transactions, as well as removing the pain of eventual consistency issues and stale reads.
- Create an always-on database that accepts reads and writes on all nodes without generating conflicts.
- Allow flexible deployment in any environment, without tying you to any platform or vendor.
- Support familiar tools for working with relational data (i.e., SQL).

With the confluence of these features, we hope that CockroachDB helps you build global, scalable, resilient deployments and applications.

Overview

CockroachDB starts running on machines with two commands:

- cockroach start with a --join flag for all of the initial nodes in the cluster, so the process knows all of the other machines it can communicate with.
- cockroach init to perform a one-time initialization of the cluster.
- Once the CockroachDB cluster is initialized, developers interact with CockroachDB through a PostgreSQL-compatible SQL API. Thanks to the symmetrical behavior of all nodes in a cluster, you can send SQL requests to any node; this makes CockroachDB easy to integrate with load balancers.

Layers

- At the highest level, CockroachDB converts clients' SQL statements into key-value (KV) data, which is distributed among nodes and written to disk. CockroachDB's architecture is manifested as a number of layers, each of which interacts with the layers directly above and below it as relatively opaque services.
- The following pages describe the function each layer performs, while mostly ignoring the details of other layers.

Layer	Order	Purpose
SQL	1	Translate client SQL queries to KV operations.
Transactional	2	Allow atomic changes to multiple KV entries.
Distribution	3	Present replicated KV ranges as a single entity.
Replication	4	Consistently and synchronously replicate KV ranges across many nodes. This layer also enables consistent reads using a consensus algorithm.
Storage	5	Read and write KV data on disk.

Database terms

Consistency

- The requirement that a transaction must change affected data only in allowed ways. CockroachDB uses "consistency" in both the sense of ACID semantics and the CAP theorem, albeit less formally than either definition.

Isolation

- The degree to which a transaction may be affected by other transactions running at the same time.
- CockroachDB provides the SERIALIZABLE isolation level, which is the highest possible and guarantees that every committed transaction has the same result as if each transaction were run one at a time.

Consensus

- The process of reaching agreement on whether a transaction is committed or aborted. CockroachDB uses the Raft consensus protocol.

In CockroachDB, when a range receives a write, a quorum of nodes containing replicas of the range acknowledge the write.

Replication

- The process of creating and distributing copies of data, as well as ensuring that those copies remain consistent.
- CockroachDB requires all writes to propagate to a quorum of copies of the data before being considered committed. This ensures the consistency of your data.

Transaction

- A set of operations performed on a database that satisfy the requirements of ACID semantics.
- This is a crucial feature for a consistent system to ensure developers can trust the data in their database.
- For more information about how transactions work in CockroachDB, see Transaction Layer.

Transaction contention

A state of conflict that occurs when:

- A transaction is unable to complete due to another concurrent or recent transaction attempting to write to the same data. This is also called lock contention.

Multi-active availability

- A consensus-based notion of high availability that lets each node in the cluster handle reads and writes for a subset of the stored data (on a per-range basis).
- This is in contrast to active-passive replication, in which the active node receives 100% of request traffic, and active-active replication, in which all nodes accept requests but typically cannot guarantee that reads are both up-to-date and fast.

User

- A SQL user is an identity capable of executing SQL statements and performing other cluster actions against CockroachDB clusters.
- SQL users must authenticate with an option permitted on the cluster (username/password, single sign-on (SSO), or certificate). Note that a SQL/cluster user is distinct from a CockroachDB Cloud organization user.

CockroachDB architecture terms

cluster

- A group of interconnected CockroachDB nodes that function as a single distributed SQL database server.
- Nodes collaboratively organize transactions, and rebalance workload and data storage to optimize performance and fault-tolerance.
- Each cluster has its own authorization hierarchy, meaning that users and roles must be defined on that specific cluster.
- A CockroachDB cluster can be run in CockroachDB Cloud, within a customer Organization, or can be self-hosted.

Node

- An individual instance of CockroachDB. One or more nodes form a cluster.

Range

- CockroachDB stores all user data (tables, indexes, etc.) and almost all system data in a sorted map of key-value pairs. This keyspace is divided into contiguous chunks called ranges, such that every key is found in one range.
- From a SQL perspective, a table and its secondary indexes initially map to a single range, where each key-value pair in the range represents a single row in the table (also called the primary index because the table is sorted by the primary key) or a single row in a secondary index. As soon as the size of a range reaches the default range size, it is split into two ranges. This process continues for these new ranges as the table and its indexes continue growing.

Replica

- A copy of a range stored on a node. By default, there are three replicas of each range on different nodes.

Leaseholder

- The replica that holds the "range lease." This replica receives and coordinates all read and write requests for the range.

Raft protocol

- The consensus protocol employed in CockroachDB that ensures that your data is safely stored on multiple nodes and that those nodes agree on the current state even if some of them are temporarily disconnected.

Raft leader

- For each range, the replica that is the "leader" for write requests. The leader uses the Raft protocol to ensure that a majority of replicas (the leader and enough followers) agree, based on their Raft logs, before committing the write. The Raft leader is almost always the same replica as the leaseholder.

Raft log

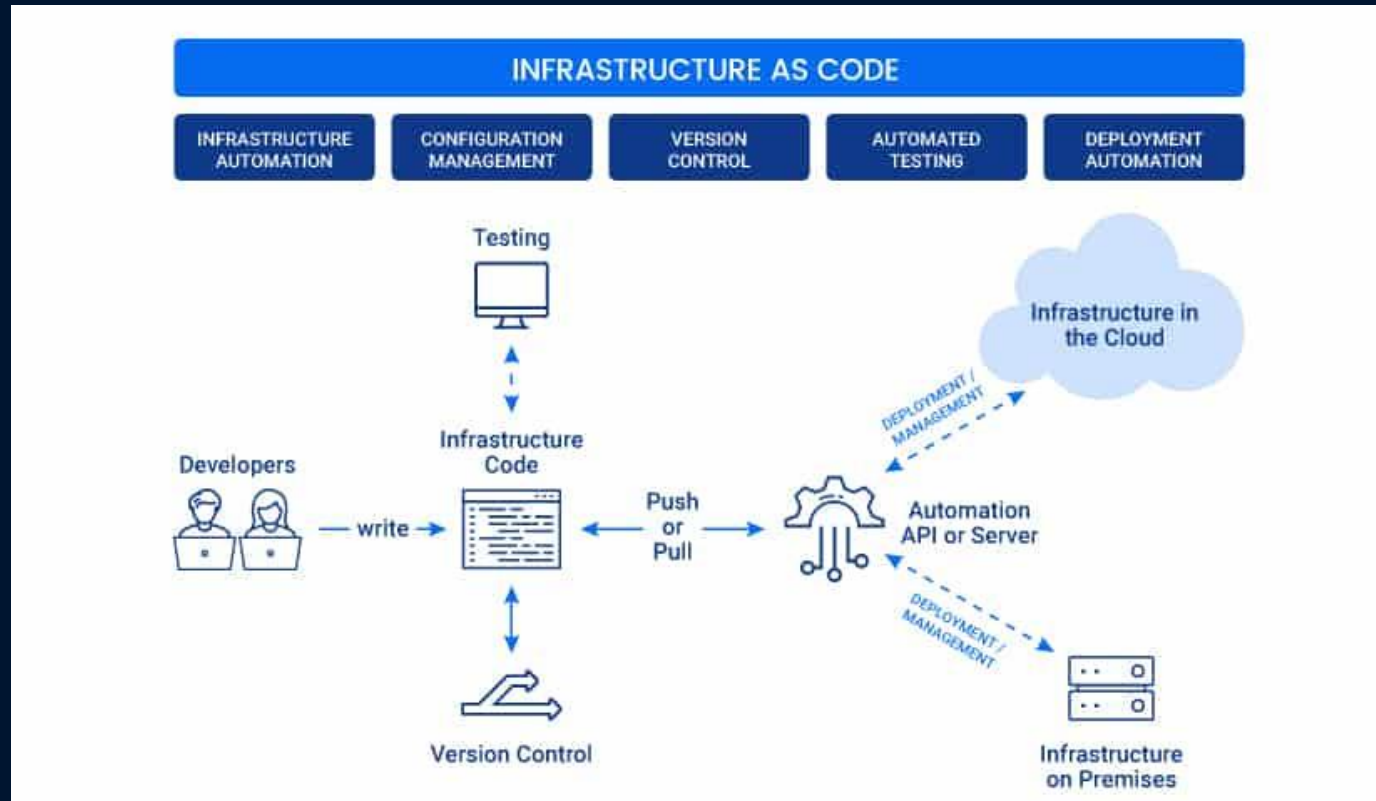
- A time-ordered log of writes to a range that its replicas have agreed on. This log exists on-disk with each replica and is the range's source of truth for consistent replication.

Getting Started with Terraform

Terraform Beginner's Guide: Everything You Should Know

What Is Infrastructure as Code (IaC)?

- Infrastructure as Code (IaC) is a widespread terminology among DevOps professionals and a key DevOps practice in the industry. It is the process of managing and provisioning the complete IT infrastructure (comprises both physical and virtual machines) using machine-readable definition files.
- It helps in automating the complete data center by using programming scripts.



Popular IaC Tools:

- Terraform An open-source declarative tool that offers pre-written modules to build and manage an infrastructure.
- Chef: A configuration management tool that uses cookbooks and recipes to deploy the desired environment. Best used for Deploying and configuring applications using a pull-based approach.
- Puppet: Popular tool for configuration management that follows a Client-Server Model. Puppet needs agents to be deployed on the target machines before the puppet can start managing them.
- Ansible: Ansible is used for building infrastructure as well as deploying and configuring applications on top of them. Best used for Ad hoc analysis.
- Packer: Unique tool that generates VM images (not running VMs) based on steps you provide. Best used for Baking compute images.
- Vagrant: Builds VMs using a workflow. Best used for Creating pre-configured developer VMs within VirtualBox.

What Is Terraform?

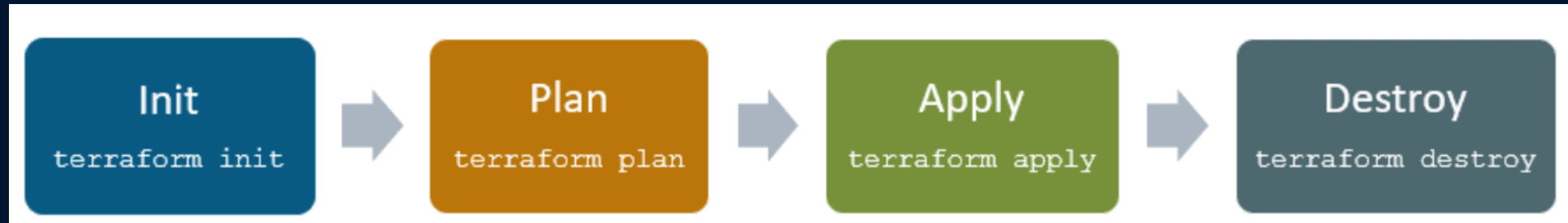
- Terraform is one of the most popular Infrastructure-as-code (IaC) tool, used by DevOps teams to automate infrastructure tasks.
- It is used to automate the provisioning of your cloud resources.
- Terraform is an open-source, cloud-agnostic provisioning tool developed by HashiCorp and written in GO language.

Benefits of using Terraform:

- Does orchestration, not just configuration management
- Supports multiple providers such as AWS, Azure, Oracle, GCP, and many more
- Provide immutable infrastructure where configuration changes smoothly
- Uses easy to understand language, HCL (HashiCorp configuration language)
- Easily portable to any other provider

Terraform Lifecycle

Terraform lifecycle consists of – init, plan, apply, and destroy.



- Terraform init initializes the (local) Terraform environment. Usually executed only once per session.
- Terraform plan compares the Terraform state with the as-is state in the cloud, build and display an execution plan. This does not change the deployment (read-only).
- Terraform apply executes the plan. This potentially changes the deployment.
- Terraform destroy deletes all resources that are governed by this specific terraform environment.

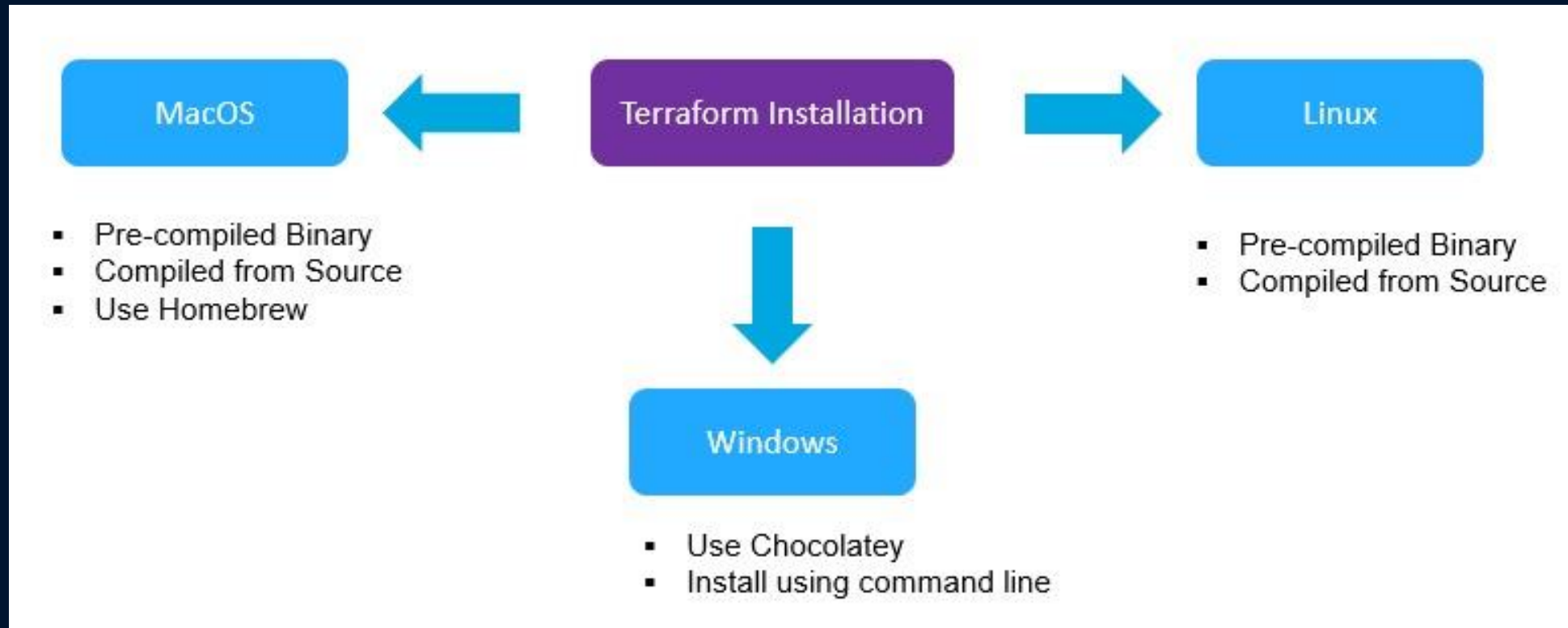
Terraform Core Concepts

- **Variables:** Terraform has input and output variables, it is a key-value pair. Input variables are used as parameters to input values at run time to customize our deployments. Output variables are return values of a terraform module that can be used by other configurations.
- **Provider:** Terraform users provision their infrastructure on the major cloud providers such as AWS, Azure, OCI, and others. A provider is a plugin that interacts with the various APIs required to create, update, and delete various resources.
- **Module:** Any set of Terraform configuration files in a folder is a module. Every Terraform configuration has at least one module, known as its root module.
- **State:** Terraform records information about what infrastructure is created in a Terraform state file. With the state file, Terraform is able to find the resources it created previously, supposed to manage and update them accordingly.

- **Resources:** Cloud Providers provides various services in their offerings, they are referenced as Resources in Terraform. Terraform resources can be anything from compute instances, virtual networks to higher-level components such as DNS records. Each resource has its own attributes to define that resource.
- **Data Source:** Data source performs a read-only operation. It allows data to be fetched or computed from resources/entities that are not defined or managed by Terraform or the current Terraform configuration.
- **Plan:** It is one of the stages in the Terraform lifecycle where it determines what needs to be created, updated, or destroyed to move from the real/current state of the infrastructure to the desired state.
- **Apply:** It is one of the stages in the Terraform lifecycle where it applies the changes real/current state of the infrastructure in order to achieve the desired state.

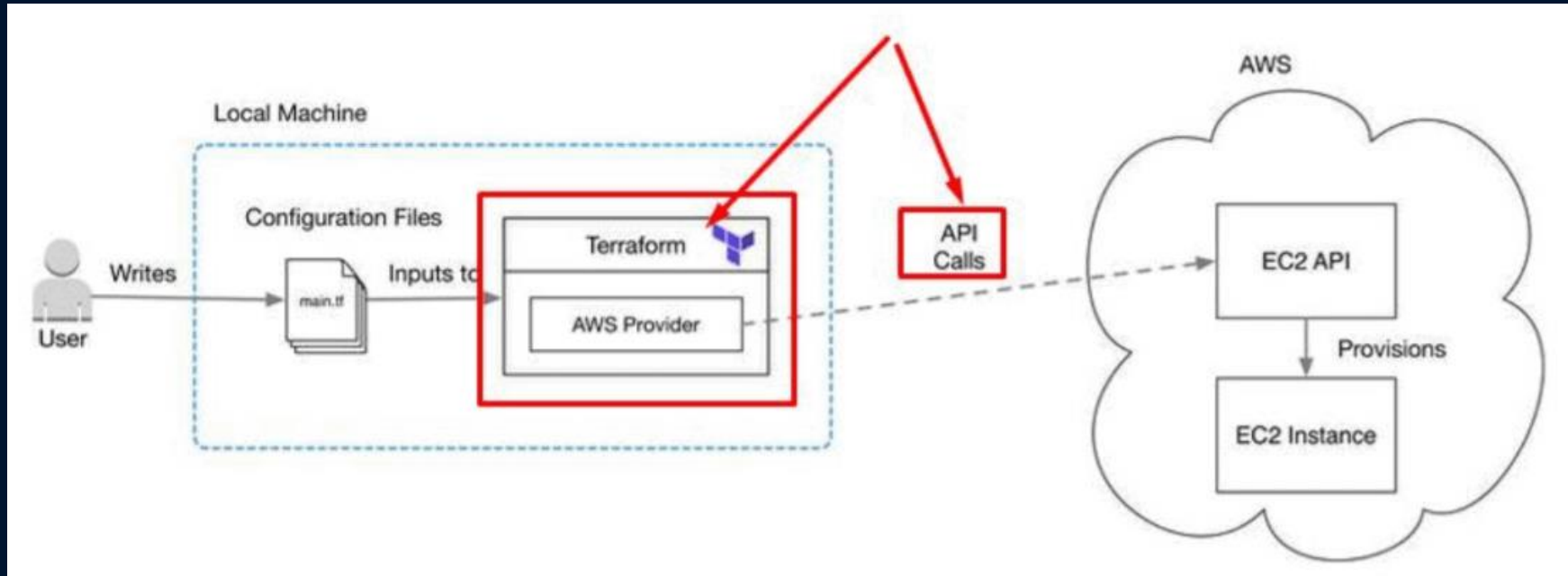
Terraform Installation

- Before you start working, make sure you have Terraform installed on your machine, it can be installed on any OS, say Windows, macOS, Linux, or others.
- Terraform installation is an easy process and can be done in a few minutes.
- Read our blog to know how to install Terraform in Linux, Mac, Windows



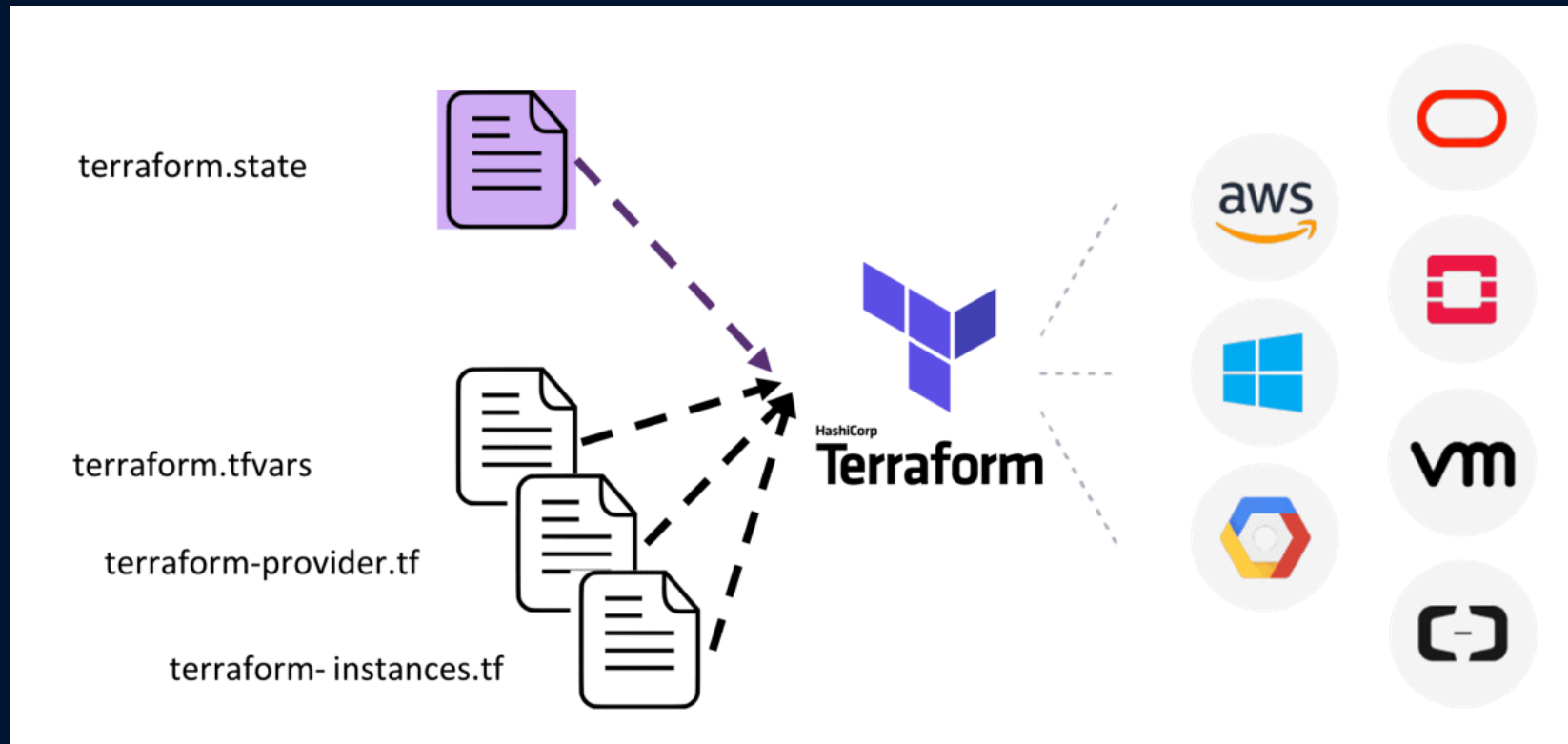
Terraform Providers

- A provider is responsible for understanding API interactions and exposing resources.
- It is an executable plug-in that contains code necessary to interact with the API of the service.
- Terraform configurations must declare which providers they require so that Terraform can install and use them.



Terraform Configuration Files

- Configuration files are a set of files used to describe infrastructure in Terraform and have the file extensions .tf and .tf.json. Terraform uses a declarative model for defining infrastructure.
- Configuration files let you write a configuration that declares your desired state. Configuration files are made up of resources with settings and values representing the desired state of your infrastructure.



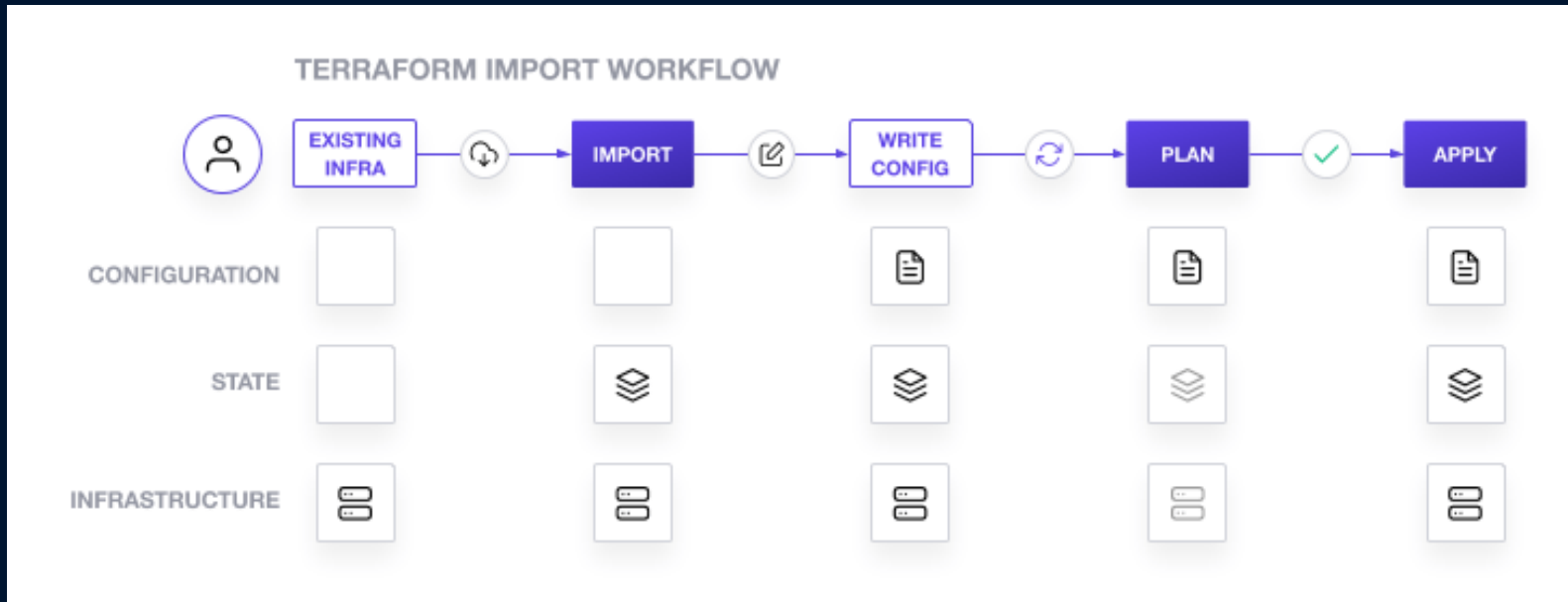
Getting started using Terraform

To get started building infrastructure resources using Terraform, there are few things that you should take care of. The general steps to deploy a resource(s) in the cloud are:

- Set up a Cloud Account on any cloud provider (AWS, Azure, OCI)
- Install Terraform
- Add a provider – AWS, Azure, OCI, GCP, or others
- Write configuration files
- Initialize Terraform Providers
- PLAN (DRY RUN) using terraform plan
- APPLY (Create a Resource) using terraform apply
- DESTROY (Delete a Resource) using terraform destroy

Import Existing Infrastructure

- Terraform is one of the great IaC tools with which, you can deploy all your infrastructure's resources.
- In addition to that, you can manage infrastructures from different cloud providers, such as AWS, Google Cloud, etc. But what if you have already created your infrastructure manually?
- Terraform has a really nice feature for importing existing resources, which makes the migration of existing infrastructure into Terraform a lot easier.



AWS Virtual Machine Using AWS Console | Terraform AWS Deployment Overview

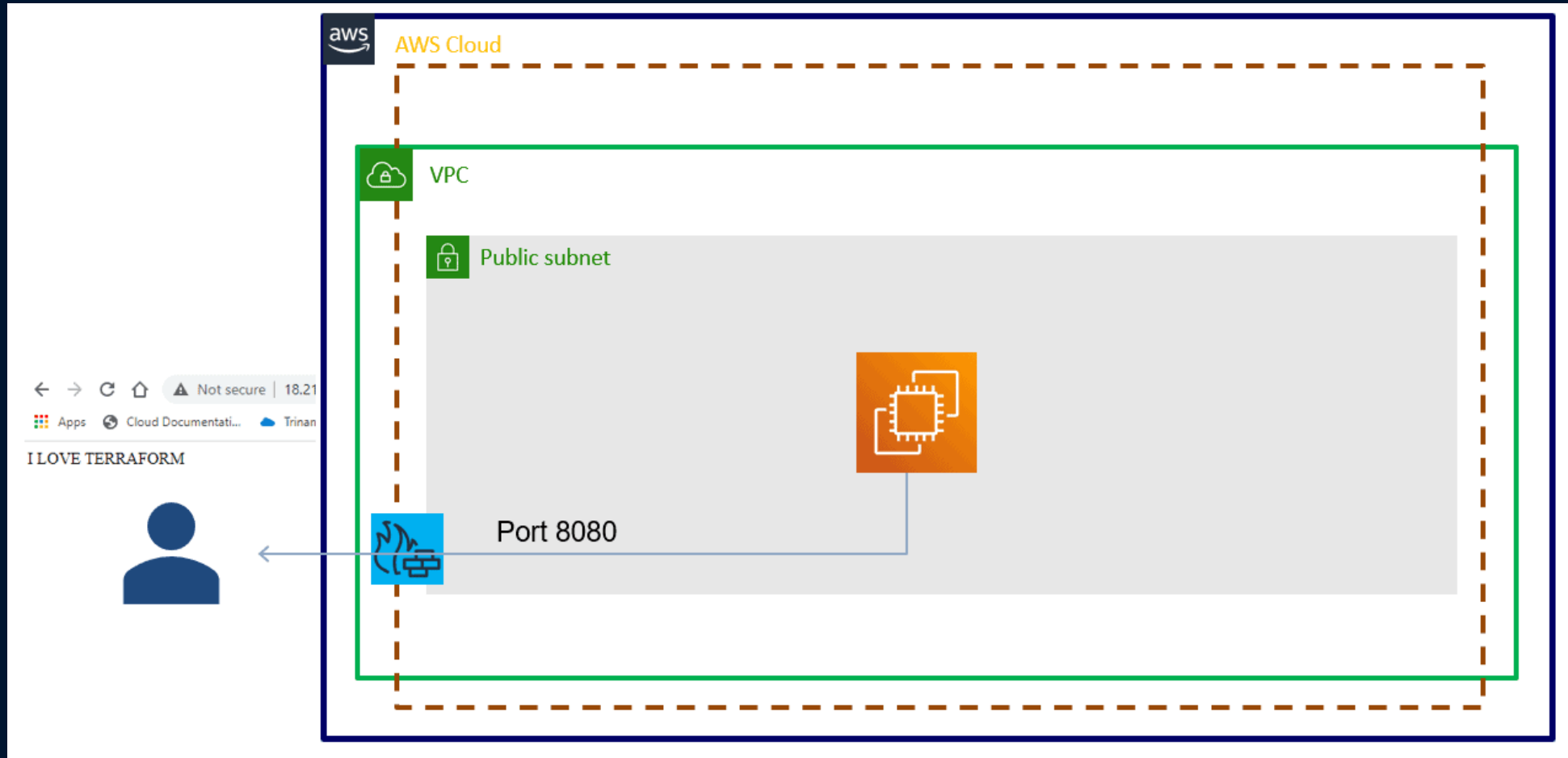
In this blog, we will see how to create an AWS Virtual machine in two different ways, one using the AWS console and the other using terraform.

Pre-requisites:

- AWS Free Tier or Paid account. Check how to create an AWS free tier account
- Terraform should be installed. Check how to install terraform

Architecture

To give a brief about what demo we will be performing, please refer to the below Architecture

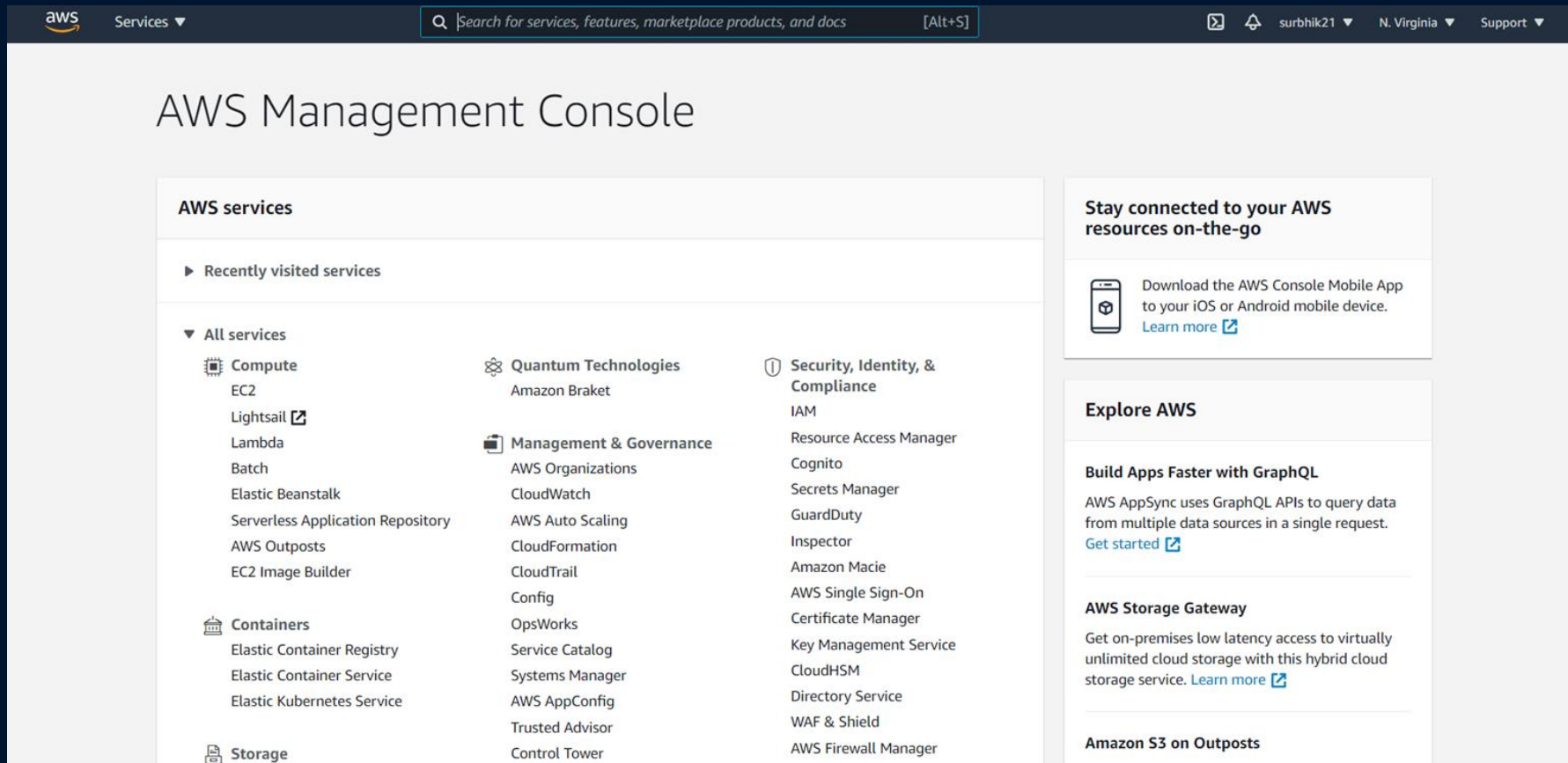


Create VM in AWS Using Terraform

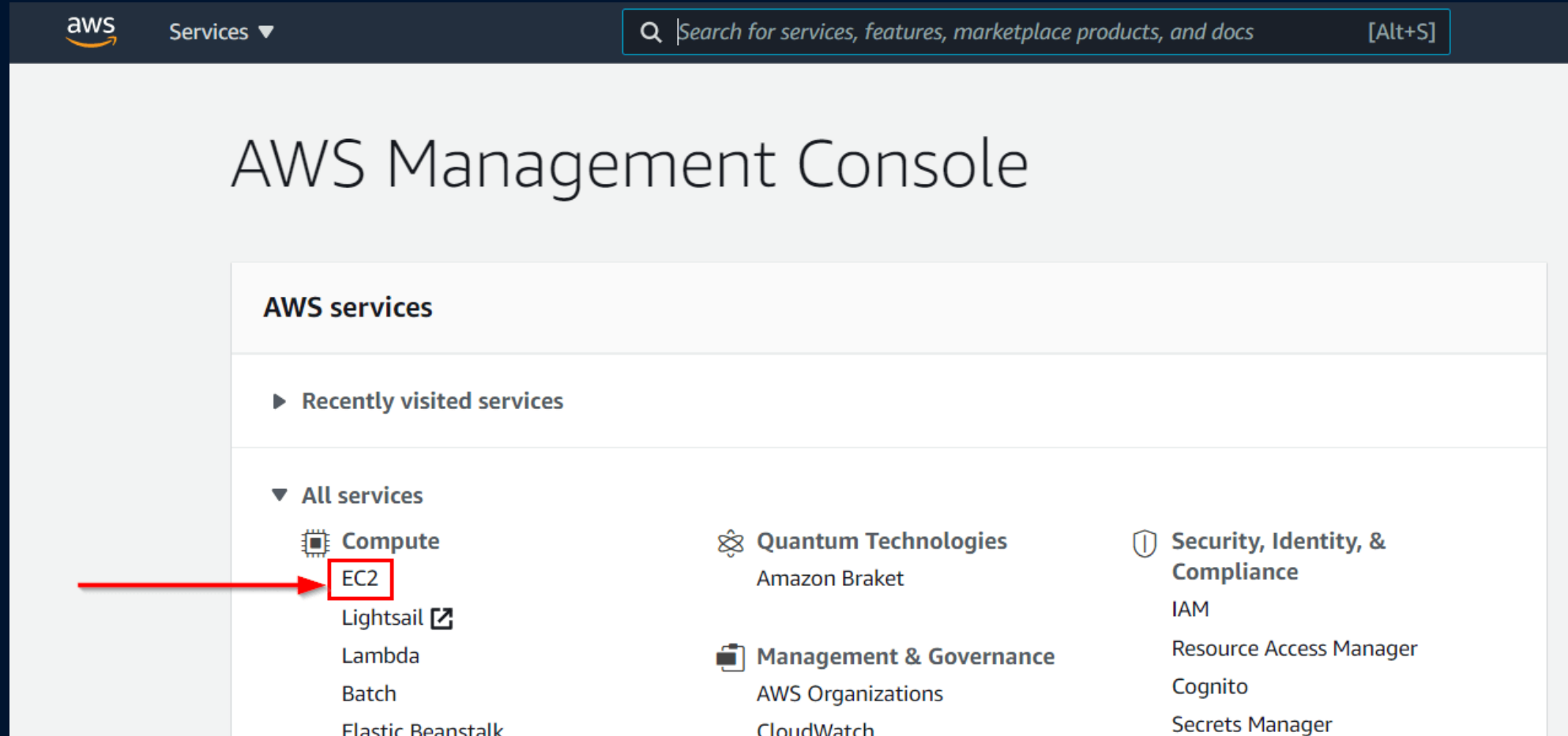
There are a series of steps that you need to follow to create an AWS Virtual Machine and deploy a webpage on top of it.

Now, below are the steps that you can follow to create the above-shown architecture:

- Log in to your AWS account.



- Go to EC2 from the navigation menu present under Compute service.



- Go to Instances from the navigation menu on the left and click on Launch instances.

The screenshot shows the AWS Management Console interface. On the left, the navigation menu has the 'Instances' link highlighted with a red box and a red arrow labeled '1'. The main content area displays a list of EC2 instances. At the top right of the main content area, the 'Launch instances' button is highlighted with a red box and a red arrow labeled '2'.

Welcome to the new instances experience!
We're redesigning the EC2 console to make it easier to use. To switch between the old console and the new console, use the New EC2 Experience toggle above the navigation panel. We'll release updates continuously based on customer feedback.

Instances (12) Info

Filter instances

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
<input type="checkbox"/>	-	i-084bb0def65d50ced	Stopped	t2.micro	-	No alarms	us-east-1a	-
<input type="checkbox"/>	-	i-023bb3454f8050b50	Stopped	t2.micro	-	No alarms	us-east-1e	-
<input type="checkbox"/>	-	i-0f3917eefa7a99e23	Running	t2.micro	2/2 checks ...	No alarms	us-east-1b	ec2-18-206-145-
<input type="checkbox"/>	Cloned_EC2	i-0fbd459c58869f7fc	Stopped	t2.micro	-	No alarms	us-east-1b	-
<input type="checkbox"/>	-	i-0df4f22e3f288ca3d	Stopped	t2.micro	-	No alarms	us-east-1b	-
<input type="checkbox"/>	Webserver	i-04983c78840fa0bc5	Stopped	t2.micro	-	No alarms	us-east-1b	-
<input type="checkbox"/>	ProdInstance	i-06a2207a0fec9215b	Stopped	t2.micro	-	No alarms	us-east-1b	-

- Search for Ubuntu and select Ubuntu Server 20.04 LTS (HTM).

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search:

Quick Start (8)

- My AMIs (0)
- AWS Marketplace (559)
- Community AMIs (35992)
- ☐ Free tier only ⓘ

AMI ID	AMI Name	Architecture	Root Device Type	Virtualization Type	ENA Enabled	Action
ami-0885b1f6bd170450c / ami-054e49cb26c2fd312	Ubuntu Server 20.04 LTS (HVM), SSD Volume Type	64-bit x86 / 64-bit Arm	ebs	hvm	Yes	Select
ami-00ddb0e5626798373 / ami-074db80f0dc9b5f40	Ubuntu Server 18.04 LTS (HVM), SSD Volume Type	64-bit x86 / 64-bit Arm	ebs	hvm	Yes	Select
ami-03e0fdb8c9d235984	Deep Learning AMI (Ubuntu 18.04) Version 39.0	64-bit x86	ebs	hvm	Yes	Select

- Select Instance type as t2.micro and click on Next: Configure Instance Details.

aws

Services ▾

Q Search for services, features, marketplace products, and docs [Alt+S]

surbhik21 ▾ N. Virginia ▾ Support ▾

1. Choose AMI

2. Choose Instance Type

3. Configure Instance

4. Add Storage

5. Add Tags

6. Configure Security Group

7. Review

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by:

All instance families ▾

Current generation ▾

Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

	Family ▾	Type ▾	vCPUs ⓘ ▾	Memory (GiB) ▾	Instance Storage (GB) ⓘ ▾	EBS-Optimized Available ⓘ ▾	Network Performance ⓘ ▾	IPv6 Support ⓘ ▾
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
<input type="checkbox"/>	t3	t3.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes

Cancel

Previous

Review and Launch

Next: Configure Instance Details

- On Configure Instance Details page, scroll down and add the following code in User data to provision the web page after the instance is provisioned. Click on 6. Configure Security Group to skip Storage and Tags.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 3: Configure Instance Details

Tenancy ⓘ Shared - Run a shared hardware instance
Additional charges will apply for dedicated tenancy.

Elastic Inference ⓘ ☐ Add an Elastic Inference accelerator
Additional charges apply.

Credit specification ⓘ ☐ Unlimited
Additional charges may apply.

File systems ⓘ

▼ Advanced Details

Enclave ⓘ ☐ Enable

Metadata accessible ⓘ Enabled

Metadata version ⓘ V1 and V2 (token optional)

Metadata token response hop limit ⓘ 1

User data ⓘ ☒ As text ☐ As file ☐ Input is already base64 encoded

1

```
#!/bin/bash
echo "I LOVE TERRAFORM" > index.html
nohup busybox http -f -p 8080 &
```

2

Cancel Previous **Review and Launch** Next: Add Storage

- Add Custom TCP protocol to open 8080 port accessible from the internet and click on Review and Launch.

aws

Services ▾

Q

Search for services, features, marketplace products, and docs

[Alt+S]

surbhik21 ▾

N. Virginia ▾

Support ▾

1. Choose AMI

2. Choose Instance Type

3. Configure Instance

4. Add Storage

5. Add Tags

6. Configure Security Group

7. Review

Step 6: Configure Security Group


A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a **new** security group
☐ Select an **existing** security group


Security group name:

Description:

Type <small>i</small>	Protocol <small>i</small>	Port Range <small>i</small>	Source <small>i</small>	Description <small>i</small>	
Custom TCP <small>f</small> ▾	TCP	8080	Custom ▾ 0.0.0.0/0	e.g. SSH for Admin Desktop	✕



Warning
You will not be able to connect to this instance as the AMI requires port(s) 22 to be open in order to have access. Your current security group doesn't have port(s) 22 open.



Warning
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

- Review all the information and click on Launch.

aws

Services

Search for services, features, marketplace products, and docs

[Alt+S]

surbhik21

N. Virginia

Support

1. Choose AMI

2. Choose Instance Type

3. Configure Instance

4. Add Storage

5. Add Tags

6. Configure Security Group

7. Review

Step 7: Review Instance Launch


Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

⚠

Improve your instances' security. Your security group, launch-wizard-36, is open to the world.
Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.
You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. [Edit security groups](#)

▼ AMI Details

Edit AMI

 **Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-0885b1f6bd170450c**

Free tier eligible

Ubuntu Server 20.04 LTS (HVM),EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).
Root Device Type: ebs Virtualization type: hvm

▼ Instance Type

Edit instance type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	-	1	1	EBS only	-	Low to Moderate

▼ Security Groups

Edit security groups

Security group name

launch-wizard-36

Description

launch-wizard-36 created 2021-01-29T17:05:56.078+05:30

Cancel

Previous

Launch

- When asked to select a key pair, select Proceed without a key pair and click on Launch Instances.

Select an existing key pair or create a new key pair

×

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

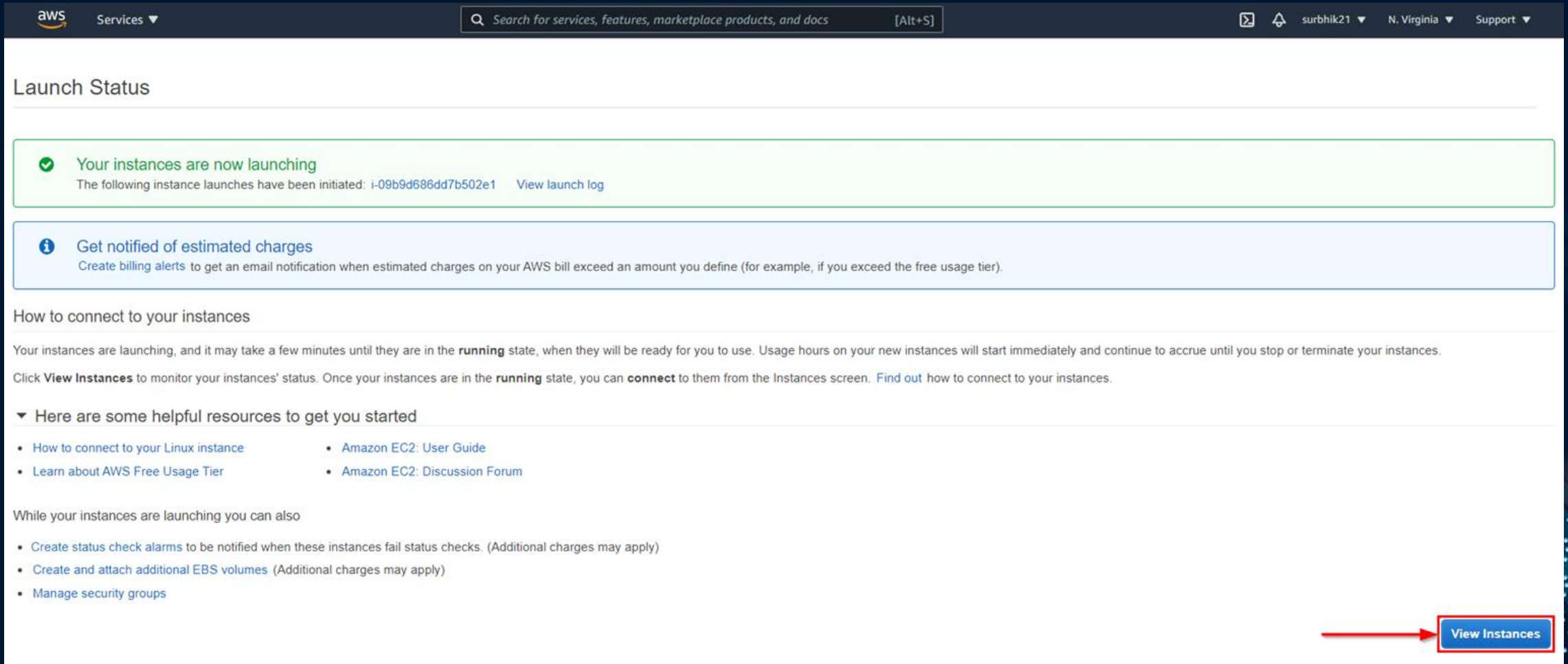
Proceed without a key pair

☒ I acknowledge that I will not be able to connect to this instance unless I already know the password built into this AMI.

Cancel

Launch Instances

- Click on View Instance to check the status of the Instance.



aws Services ▾

Search for services, features, marketplace products, and docs [Alt+S]

surbhik21 ▾ N. Virginia ▾ Support ▾

Launch Status

✓ **Your instances are now launching**

The following instance launches have been initiated: [i-09b9d686dd7b502e1](#) [View launch log](#)

i **Get notified of estimated charges**

Create [billing alerts](#) to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click **View Instances** to monitor your instances' status. Once your instances are in the **running** state, you can **connect** to them from the Instances screen. [Find out](#) how to connect to your instances.

▼ Here are some helpful resources to get you started

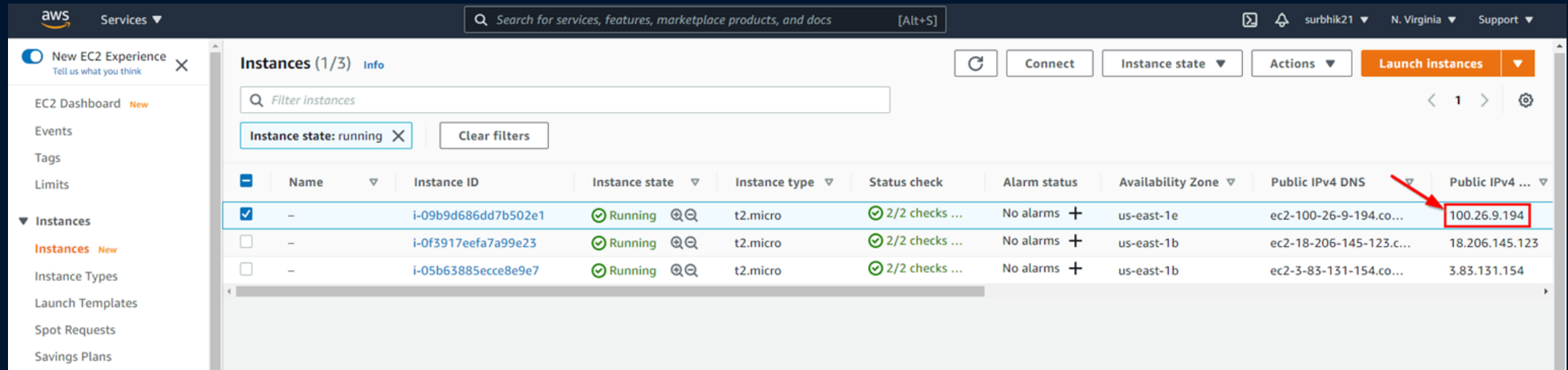
- [How to connect to your Linux instance](#)
- [Amazon EC2: User Guide](#)
- [Learn about AWS Free Usage Tier](#)
- [Amazon EC2: Discussion Forum](#)

While your instances are launching you can also

- [Create status check alarms](#) to be notified when these instances fail status checks. (Additional charges may apply)
- [Create and attach additional EBS volumes](#) (Additional charges may apply)
- [Manage security groups](#)

[View Instances](#)

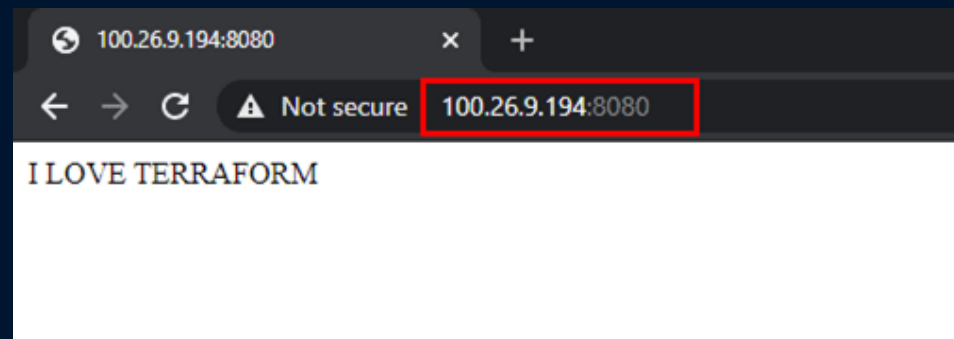
- Once the instance is up and running, copy the Public IP.



The screenshot shows the AWS Management Console 'Instances' page. The table lists three running EC2 instances. The first instance, with ID i-09b9d686dd7b502e1, is selected. Its Public IPv4 address, 100.26.9.194, is highlighted with a red box and a red arrow.

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
<input checked="" type="checkbox"/>	-	i-09b9d686dd7b502e1	Running	t2.micro	2/2 checks ...	No alarms	us-east-1e	ec2-100-26-9-194.co...	100.26.9.194
<input type="checkbox"/>	-	i-0f3917eefa7a99e23	Running	t2.micro	2/2 checks ...	No alarms	us-east-1b	ec2-18-206-145-123.c...	18.206.145.123
<input type="checkbox"/>	-	i-05b63885ecce8e9e7	Running	t2.micro	2/2 checks ...	No alarms	us-east-1b	ec2-3-83-131-154.co...	3.83.131.154

- Paste the IP in your browser followed by port 8080 to access the created webpage.



Terraform AWS Deployment

We will use the below code to provision AWS Virtual Machine in our account. You can download the code by clicking [here](#).

```
provider "aws" {  
  region = "us-east-1"  
  access_key = "<YOUR ACCESS KEY HERE>"  
  secret_key = "<YOUR SECRET KEY HERE>"  
}  
resource "aws_instance" "tfvm" {  
  ami = "ami-0885b1f6bd170450c"  
  instance_type = "t2.micro"  
  vpc_security_group_ids = [ aws_security_group.websg.id ]  
  user_data = <<-EOF  
#!/bin/bash  
    echo "I LOVE TERRAFORM" > index.html  
    nohup busybox httpd -f -p 8080 &  
    EOF
```

```
tags = {  
    Name = "WEB-demo"  
}  
}  
resource "aws_security_group" "websg" {  
    name = "web-sg01"  
    ingress {  
        protocol = "tcp"  
        from_port = 8080  
        to_port = 8080  
        cidr_blocks = [ "0.0.0.0/0" ]  
    }  
}  
output "instance_ips" {  
    value = aws_instance.tfvm.public_ip  
}
```

Steps to create the VM from the code:

- Use terraform init command in terminal to initialize terraform and download the configuration files

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\HAWX\Documents\terraform-awsvm> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.26.0...
- Installed hashicorp/aws v3.26.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\Users\HAWX\Documents\terraform-awsvm> |
```


- Now let's check for any human error in our script by using terraform plan command.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\HAWX\Documents\terraform-awsvm> terraform plan

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

# aws_instance.tfvm will be created
+ resource "aws_instance" "tfvm" {
  + ami                  = "ami-0885b1f6bd170450c"
  + arn                  = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone     = (known after apply)
  + cpu_core_count        = (known after apply)
  + cpu_threads_per_core   = (known after apply)
  + get_password_data      = false
  + host_id               = (known after apply)
  + id                   = (known after apply)
  + instance_state         = (known after apply)
  + instance_type          = "t2.micro"
  + ipv6_address_count     = (known after apply)
  + ipv6_addresses         = (known after apply)
  + key_name               = (known after apply)
  + outpost_arn            = (known after apply)
  + password_data          = (known after apply)
  + placement_group        = (known after apply)
  + primary_network_interface_id = (known after apply)
  + private_dns             = (known after apply)
  + private_ip             = (known after apply)
  + public_dns             = (known after apply)
  + public_ip              = (known after apply)
  + secondary_private_ips   = (known after apply)
```

In the end, it will also show us the number of resources that will be added to our AWS account.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

      + prefix_list_ids = []
      + protocol        = "tcp"
      + security_groups = []
      + self            = false
      + to_port          = 8080
    },
  ]
+ name          = "web-sg01"
+ owner_id      = (known after apply)
+ revoke_rules_on_delete = false
+ vpc_id        = (known after apply)
}

Plan: 2 to add, 0 to change, 0 to destroy.

-----

Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.

PS C:\Users\HAWX\Documents\terraform-awsvm>
```

- Now we will apply our code using terraform apply -auto-approve command. It will provide us the IP of our created VM.

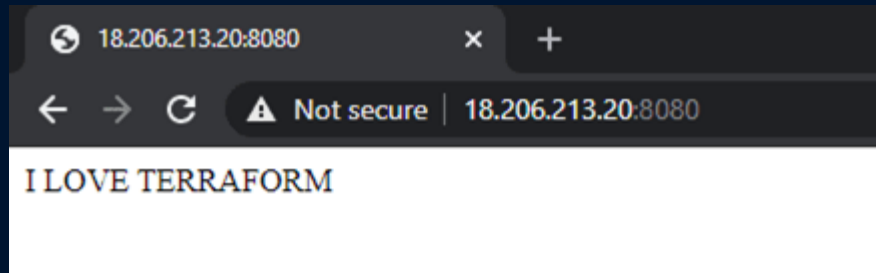
```
PS C:\Users\HAWX\Documents\terraform-awsvm> terraform apply -auto-approve
aws_security_group.websg: Creating...
aws_security_group.websg: Creation complete after 8s [id=sg-019465038b95617b1]
aws_instance.tfvm: Creating...
aws_instance.tfvm: Still creating... [10s elapsed]
aws_instance.tfvm: Still creating... [20s elapsed]
aws_instance.tfvm: Still creating... [30s elapsed]
aws_instance.tfvm: Still creating... [40s elapsed]
aws_instance.tfvm: Creation complete after 42s [id=i-04eec7b6688fa5a04]
```

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

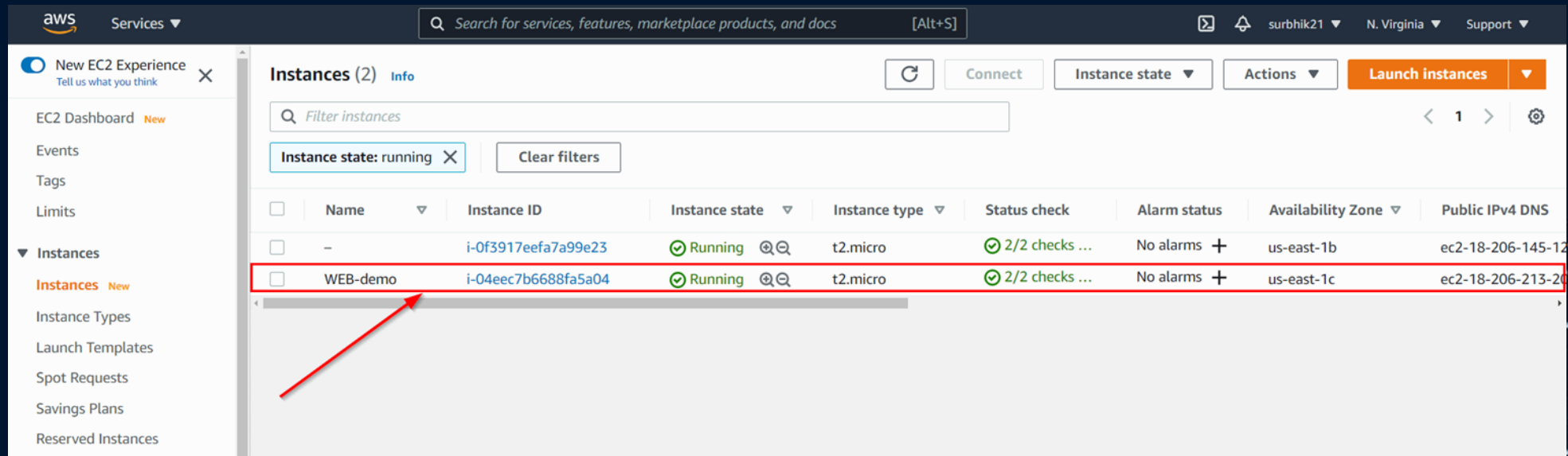
Outputs:

```
instance_ips = "18.206.213.20"
```


Now, we can check if the Webpage has been created or not. Just copy the IP and check on 8080 port regarding the same.



You can also check if the EC2 instance is created in the AWS by going to the EC2 dashboard.



Initial Deployment of CockroachDB with Terraform

Provision a CockroachDB Cloud Cluster with Terraform

- Terraform is an infrastructure-as-code provisioning tool that uses configuration files to define application and network resources.
- You can provision CockroachDB Cloud clusters and cluster resources by using the CockroachDB Cloud Terraform provider in your Terraform configuration files.
- This shows you how to provision a CockroachDB Cloud cluster using the CockroachDB Cloud Terraform provider.

CockroachDB Serverless

Before you begin

Before you start this tutorial, you must

- Install Terraform.
- Install the wget command line utility.
- Create a service account and API key in the CockroachDB Cloud Console, and assign admin privilege or Cluster Creator / Cluster Admin role at the organization scope. Refer to: Service Accounts

Create the Terraform configuration files

- In a terminal create a new directory and use wget to download the CockroachDB Serverless main.tf example file:

```
wget https://raw.githubusercontent.com/cockroachdb/terraform-provider-cockroach/main/examples/workflows/cockroach\_serverless\_cluster/main.tf
```

- In a text editor create a new file terraform.tfvars with the following settings:

```
cluster_name = "{cluster name}"
```

```
sql_user_name = "{SQL user name}"
```

```
sql_user_password = "{SQL user password}"
```

- Create an environment variable named COCKROACH_API_KEY. Copy the API key from the CockroachDB Cloud console and create the COCKROACH_API_KEY environment variable:
export COCKROACH_API_KEY={API key}

Provision the cluster

- Initialize the provider:
terraform init –upgrade
- Create the Terraform plan. This shows the actions the provider will take, but won't perform them:
terraform plan
- Create the cluster:
terraform apply

Delete the cluster

- If you want to delete the cluster, run the following command:
terraform destroy
- Enter yes when prompted to delete the cluster.

Provision a CockroachDB Cloud Cluster with Terraform

- Terraform is an infrastructure-as-code provisioning tool that uses configuration files to define application and network resources.
- You can provision CockroachDB Cloud clusters and cluster resources by using the CockroachDB Cloud Terraform provider in your Terraform configuration files.
- This tutorial shows you how to provision a CockroachDB Cloud cluster using the CockroachDB Cloud Terraform provider.

CockroachDB Dedicated

Before you begin

Before you start this tutorial, you must

- Install Terraform.
- Install the wget command line utility.
- Create a service account and API key in the CockroachDB Cloud Console, and assign admin privilege or Cluster Creator / Cluster Admin role at the organization scope. Refer to: Service Accounts

Create the Terraform configuration files

- In a terminal create a new directory and use wget to download the CockroachDB Serverless main.tf example file:

```
wget https://raw.githubusercontent.com/cockroachdb/terraform-provider-cockroach/main/examples/workflows/cockroach\_dedicated\_cluster/main.tf
```

- In a text editor create a new file terraform.tfvars with the following settings:

```
cluster_name = "{cluster name}"
sql_user_name = "{SQL user name}"
sql_user_password = "{SQL user password}"
cloud_provider = "{cloud provider}"
cloud_provider_regions = ["{cloud provider region}"]
cluster_node_count = {number of nodes}
storage_gib = {storage in GiB}
machine_type = "{cloud provider machine type}"
allow_list_name = "{allow list name}"
cidr_ip = "{allow list CIDR IP}"
cidr_mask = {allow list CIDR mask}
```


- Create an environment variable named COCKROACH_API_KEY.
- Copy the API key from the CockroachDB Cloud console and create the COCKROACH_API_KEY environment variable:

```
export COCKROACH_API_KEY={API key}
```

Provision the cluster

- **Initialize the provider:**

`terraform init -upgrade`

This reads the main.tf configuration file and uses the terraform.tfvars file for settings specific to your cluster. The -upgrade flag ensures you are using the latest version of the provider.

- **Create the Terraform plan. This shows the actions the provider will take, but won't perform them:**

`terraform plan`

- **Create the cluster:**

`terraform apply`

Enter yes when prompted to apply the plan and create the cluster.

Get information about your cluster

- The terraform show command shows detailed information of your cluster resources.
- terraform show

Delete the cluster

- If you want to delete the cluster, run the following command:
- terraform destroy

Basic Operations in CockroachDB

Exploring the CockroachDB interface

SQL Statements

CockroachDB supports the following SQL statements.

In the cockroach SQL shell, use `\h [statement]` to get inline help about a statement.

- **Data definition statements**
- **Data manipulation statements**
- **Data control statements**
- **Transaction control statements**
- **Session management statements**

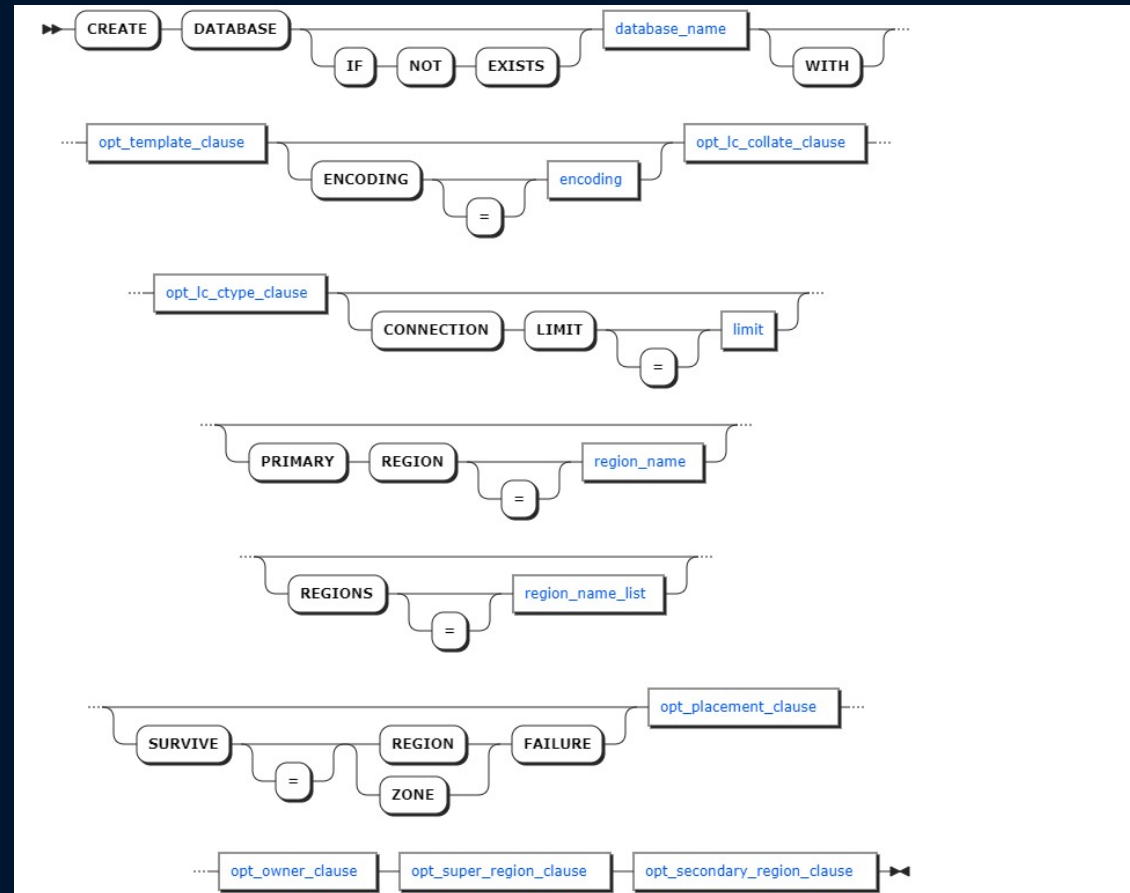
- **Cluster management statements**
- **Query management statements**
- **Query planning statements**
- **Job management statements**
- **Backup and restore statements**
- **Changefeed statements**
- **External resource statements**

CREATE DATABASE

- Required privileges

To create a database, the user must be a member of the admin role or must have the CREATEDB parameter set.

Synopsis



Example

- **Create a database**
- CREATE DATABASE bank;
- CREATE DATABASE
- SHOW DATABASES;

database_name	owner	primary_region	regions	survival_goal
bank	demo	NULL	{ }	NULL
defaultdb	root	NULL	{ }	NULL
postgres	root	NULL	{ }	NULL
system	node	NULL	{ }	NULL

(4 rows)

- Create fails (name already in use)
- **CREATE DATABASE bank;**
- ERROR: database "bank" already exists
- SQLSTATE: 42P04
- **CREATE DATABASE IF NOT EXISTS bank;**
- CREATE DATABASE

SHOW DATABASES;

database_name	owner	primary_region	regions	survival_goal
bank	demo	NULL	{ }	NULL
defaultdb	root	NULL	{ }	NULL
postgres	root	NULL	{ }	NULL
system	node	NULL	{ }	NULL

(4 rows)

Create a multi-region database

- cockroach demo --nodes=6 --demo-locality=region=us-east1,zone=us-east1-a:region=us-east1,zone=us-east1-b:region=us-central1,zone=us-central1-a:region=us-central1,zone=us-central1-b:region=us-west1,zone=us-west1-a:region=us-west1,zone=us-west1-b --no-example-database
- SHOW REGIONS;

region	zones	database_names	primary_region_of
us-central1	{us-central1-a,us-central1-b}	{}	{}
us-east1	{us-east1-a,us-east1-b}	{}	{}
us-west1	{us-west1-a,us-west1-b}	{}	{}

(3 rows)

- CREATE DATABASE bank PRIMARY REGION "us-east1" REGIONS "us-east1", "us-central1", "us-west1" SURVIVE REGION FAILURE;
- SHOW DATABASES;

database_name	owner	primary_region	regions	survival_goal
bank	demo	us-east1	{us-central1,us-east1,us-west1}	region
defaultdb	root	NULL	{}	NULL
postgres	root	NULL	{}	NULL
system	node	NULL	{}	NULL

(4 rows)

- SHOW REGIONS FROM DATABASE bank;

database	region	primary	zones
bank	us-east1	true	{us-east1-a,us-east1-b}
bank	us-central1	false	{us-central1-a,us-central1-b}
bank	us-west1	false	{us-west1-a,us-west1-b}

(3 rows)

Create a multi-region database with a secondary region

- To add a secondary region during database creation, use the following steps:
- Start a cockroach demo cluster as described in the example Create a multi-region database.
- Issue a CREATE DATABASE statement like the following. It is the same as in the Create a multi-region database example, except that it adds a SECONDARY REGION {region} clause:
- ```
CREATE DATABASE bank PRIMARY REGION "us-east1" REGIONS "us-east1", "us-central1", "us-west1" SURVIVE REGION FAILURE SECONDARY REGION "us-west1";
```
- CREATE DATABASE



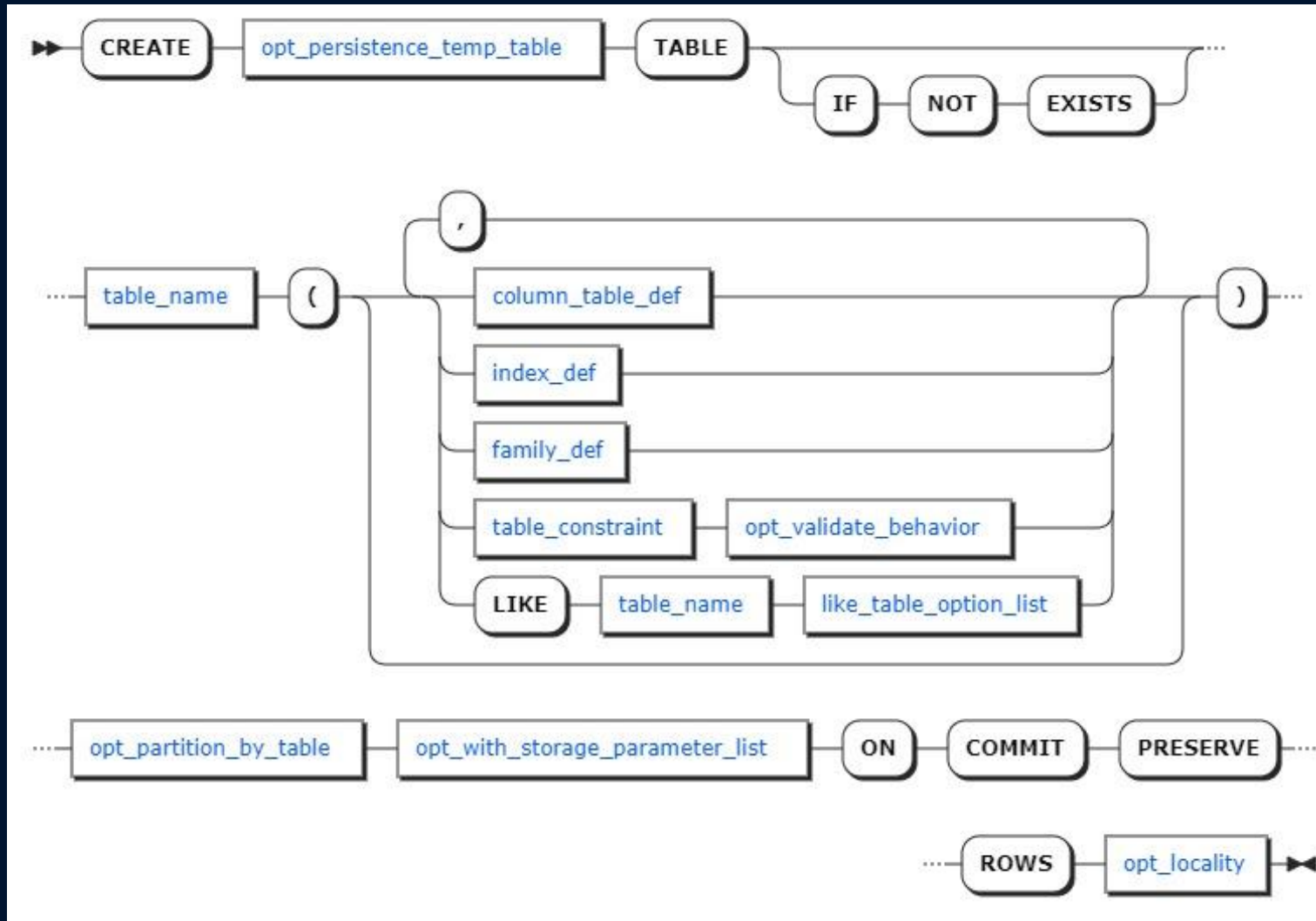
# CREATE TABLE

## Required privileges

To create a table, the user must have one of the following:

- Membership to the admin role for the cluster.
- Membership to the owner role for the database.
- The CREATE privilege on the database.

## Synopsis



## Column qualifications

CockroachDB supports the following column qualifications:

- Column-level constraints
- Collations
- Column family assignments
- DEFAULT expressions
- ON UPDATE expressions
- Identity columns (sequence-populated columns)
- NOT VISIBLE



## Create a table like an existing table

- CockroachDB supports the CREATE TABLE LIKE syntax for creating a new table based on the schema of an existing table.

The following options are supported:

- INCLUDING CONSTRAINTS adds all CHECK constraints from the source table.
- INCLUDING DEFAULTS adds all DEFAULT column expressions from the source table.
- INCLUDING GENERATED adds all computed column expressions from the source table.
- INCLUDING INDEXES adds all indexes from the source table.
- INCLUDING ALL includes all of the specifiers above.

## Create a table

In this example, we create the users table with a single primary key column defined. In CockroachDB, every table requires a primary key. If one is not explicitly defined, a column called rowid of the type INT is added automatically as the primary key, with the unique\_rowid() function used to ensure that new rows always default to unique rowid values. The primary key is automatically indexed.

## Create a table with secondary and GIN indexes

We also have other resources on indexes:

- Create indexes for existing tables using CREATE INDEX.
- Learn more about indexes.

## Create a table with auto-generated unique row IDs

- To auto-generate unique row identifiers, you can use the `gen_random_uuid()`, `uuid_v4()`, or `unique_rowid()` functions.

## Create a table with a foreign key constraint

- Foreign key constraints guarantee a column uses only values that already exist in the column it references, which must be from another table. This constraint enforces referential integrity between the two tables.
- There are a number of rules that govern foreign keys, but the most important rule is that referenced columns must contain only unique values. This means the `REFERENCES` clause must use exactly the same columns as a primary key or unique constraint.
- You can include a foreign key action to specify what happens when a column referenced by a foreign key constraint is updated or deleted. The default actions are `ON UPDATE NO ACTION` and `ON DELETE NO ACTION`.

## Create a table that mirrors key-value storage

CockroachDB is a distributed SQL database built on a transactional and strongly-consistent key-value store. Although it is not possible to access the key-value store directly, you can mirror direct access using a "simple" table of two columns, with one set as the primary key:

```
CREATE TABLE kv (k INT PRIMARY KEY, v BYTES);
```

- When such a "simple" table has no indexes or foreign keys, INSERT/UPSERT/UPDATE/DELETE statements translate to key-value operations with minimal overhead (single digit percent slowdowns).
- For example, the following UPSERT to add or replace a row in the table would translate into a single key-value Put operation:

```
UPSERT INTO kv VALUES (1, b'hello')
```

- This SQL table approach also offers you a well-defined query language, a known transaction model, and the flexibility to add more columns to the table if the need arises.



## Create a table from a SELECT statement

- You can use the CREATE TABLE AS statement to create a new table from the results of a SELECT statement.
- For example, suppose you have a number of rows of user data in the users table, and you want to create a new table from the subset of users that are located in New York.

## Create a table with a hash-sharded primary index

- We discourage indexing on sequential keys. If a table must be indexed on sequential keys, use hash-sharded indexes.
- Hash-sharded indexes distribute sequential traffic uniformly across ranges, eliminating single-range hot spots and improving write performance on sequentially-keyed indexes at a small cost to read performance.

## Create a table with a hash-sharded secondary index

- Let's now create the events table and add a secondary index on the ts column in a single statement:

## Create a new table from an existing one

- Create a table including all supportspecifiersed source
- Create a table in a multi-region database
- Create a table with a global locality
- Create a table with a regional-by-table locality
- Create a table with a regional-by-row locality
- Create a table with a regional-by-row locality, using a custom region column

## Create a table with an identity column

- Identity columns define a sequence from which to populate a column when a new row is inserted.

## Create a table with data excluded from backup

- In some situations, you may want to exclude a table's row data from a backup.
- For example, a table could contain high-churn data that you would like to garbage collect more quickly than the incremental backup schedule for the database or cluster that will hold the table. You can use the `exclude_data_from_backup = true` parameter with `CREATE TABLE` to mark a table's row data for exclusion from a backup



# Introduction to CockroachDB Clusters



# Manage a CockroachDB Serverless Cluster

## CockroachDB Serverless

### View Clusters page

On logging in to the CockroachDB Cloud Console, the Clusters page is displayed. The Clusters page provides a high-level view of your clusters.

For each cluster, the following details display:

- The cluster's Name
- The cluster's Plan type, either Serverless or Dedicated
- The date and time the cluster was Created
- The cluster's current State
- The cluster's cloud provider, either GCP or AWS
- The Version of CockroachDB the cluster is running
- The Action button, which is used to:
  - Edit resource limits
  - Delete cluster

## View cluster overview

The Overview page displays details about the selected CockroachDB Serverless cluster:

- The Cluster settings section, including Cloud provider, Plan type, and Regions
- The Usage this month section, including the Resource limits, Storage, and Request Units
- The cluster's Current activity
- Time-series graphs of the cluster's Storage usage, Request Units, and SQL statements

## Estimate usage cost

The monthly cost estimate is calculated using simple extrapolation that assumes your workload during the selected time frame is an accurate representation of your workload over the month. If you haven't been running a workload for at least the length of the selected time frame, your results will be inaccurate.

- In the Usage this month section of your cluster's Overview page, click Estimate usage cost.
- Select a time period in which your workload was active.

## Edit your resource limits

On the Overview page, you can edit your resource limits. Changes apply to the current and future billing cycles. For more details, refer to [Plan a CockroachDB Serverless cluster](#).

- Navigate to the Overview page for the cluster you want to edit.
- Click the pencil icon (or Add resource limits if you haven't set one before) next to your Resource limits in the Usage this month section.
- You will be taken to the Edit cluster page, which shows a graph of your cluster's Recommended budget compared to your current budget.
- Enter new Resource limits.
- Click Update.

## Edit regions

### ➤ Add regions

To add regions to your cluster:

- Navigate to the cluster's Overview page.
- In the Cluster settings section, click the pencil icon next to the cluster's Regions.
- The Edit cluster page displays.
- Click Add region.
- Choose the region you want to add or use the suggested one.
- In the Summary sidebar, verify the hourly estimated cost for the cluster.
- Click Update.



## ➤ **Edit the primary region**

To set the primary region:

- Navigate to the cluster's Overview page.
- In the Cluster settings section, click the pencil icon next to the cluster's Regions.
- The Edit cluster page displays.
- Select Set primary region next to your preferred region.
- Click Update.

## **Restore data from a backup**

- Use the Managed-Service Backups to restore your cluster from automatic full cluster backups:
- You can also back up and restore your CockroachDB Serverless cluster manually. You can take backups locally to userfile or back up to cloud storage.

## Delete cluster

Proceed with the following steps only if you are sure you want to delete a cluster:

- Navigate to the Overview page for the cluster you want to delete.
- Click the Actions button in the top right corner.
- Select Delete cluster.
- In the confirmation window, enter the name of the cluster.
- Click Delete.

# Manage a CockroachDB Dedicated Cluster

## CockroachDB Dedicated

### Planning your cluster

- Before making any changes to your cluster's nodes or regions, review the requirements and recommendations for CockroachDB Cloud cluster configuration.

## View Clusters page

The Clusters page provides a high-level view of your clusters.

For each cluster, the following details display:

- The cluster's Name
- The cluster's Plan Type (Serverless, Dedicated standard, or Dedicated advanced)
- The date and time the cluster was Created
- The cluster's current State
- The cluster's Cloud provider, GCP, AWS, or Azure
- The Version of CockroachDB the cluster is running
- The Action button, which is used to:
  - Add or remove nodes
  - Increase storage
  - Change compute
  - Upgrade major version
  - Delete cluster



## View cluster overview

The Overview page displays details about the selected CockroachDB Cloud cluster:

The cluster's Configuration shows details about the cluster, its deployment environment, and its nodes, such as the cluster's cloud provider, plan type, regions, and each node's status, compute, and storage.

The Cluster upgrades section shows the cluster's Upgrade window for patch upgrades and the current value for the Delay patch upgrades setting.

- The PCI Ready section shows the status of features required for PCI DSS. Requires CockroachDB Dedicated advanced.
- The status of security features required for PCI readiness.

## Scale your cluster

These sections show how to scale a Dedicated cluster horizontally by adding or removing nodes or vertically by changing each node's storage and compute resources.

### ➤ Add or remove nodes from a cluster

To add or remove nodes from your cluster:

- Navigate to the cluster's Overview page.
- Select Actions > Edit cluster.
- The Edit cluster page displays.
- From the Nodes dropdown, select the number of nodes you want in each region.
- In the Summary sidebar, verify the hourly estimated cost for the cluster.
- Click Next: Payment.
- On the Summary page, verify your new cluster configuration.
- Click Update.

## ➤ Increase storage for a cluster

- Navigate to the cluster's Overview page.
- Select Actions > Edit cluster.
- The Edit cluster page displays.
- Navigate to the Storage dropdown in the Hardware per node section.
- Select the new amount of storage per node.
- In the Summary sidebar, verify the hourly estimated cost for the cluster.
- Click Next: Payment.
- On the Summary page, verify your new cluster configuration.
- Click Update.

## ➤ Change compute for a cluster

- Navigate to the cluster's Overview page.
- Select Actions > Edit cluster.
- The Edit cluster page displays.
- Navigate to the Compute dropdown in the Hardware per node section.
- Select the new amount of vCPUs per node.
- In the Summary sidebar, verify the hourly estimated cost for the cluster.
- Click Next: Payment.
- On the Summary page, verify your new cluster configuration.
- Click Update.



## Add or remove regions from a cluster

### ➤ Add a region to your cluster

You can add up to nine regions at a time through the Console. See Planning your cluster for cluster requirements and recommendations before proceeding.

- Navigate to the cluster's Overview page.
- Select Actions > Edit cluster.
- The Edit cluster page displays.
- Click Add a region.
- If you have a GCP cluster with VPC peering enabled, the IP range will be automatically populated for added regions.
- In the Regions & Nodes section, select the desired new region and specify the number of nodes for it.
- In the Summary sidebar, verify the hourly estimated cost for the cluster.
- Click Update.

## ➤ **Remove a region from your cluster**

To remove a region from your cluster:

- Navigate to the cluster's Overview page.
- Select Actions > Edit cluster.
- The Edit cluster page displays.
- Click the X button next to each region you want to remove.
- In the Summary sidebar, verify the hourly estimated cost for the cluster.
- Click Update.

## Set a maintenance window

To set a maintenance window:

- Click the pencil icon next to Cluster maintenance to edit the maintenance window.
- From the Day dropdown, select the day of the week during which maintenance may be applied.
- From the Start of window dropdown, select a start time for your maintenance window in UTC.
- The window will last for 6 hours from the start time.
- (Optional) If you want to delay automatic patch upgrades for 60 days, switch Delay patch upgrades to On.

Enable this setting for production clusters to ensure that development and testing clusters are upgraded before production clusters. This setting applies only to patch versions and not to other kinds of upgrades.

## Restore data from a backup

- Cockroach Labs runs full backups daily and incremental backups hourly for every CockroachDB Dedicated cluster. Full backups are retained for 30 days and incremental backups for 7 days. See the [Use Managed-Service Backups](#) page for ways to restore data from your cluster's automatic backups in the Console.

## Configure PCI ready features (Dedicated advanced)

CockroachDB Dedicated advanced clusters have a PCI ready panel to monitor the status of security features required for PCI readiness. Feature statuses will update from INACTIVE to ACTIVE once you configure them. Learn more about configuring these features:

- CockroachDB Cloud Organization Audit logs
- Customer-Managed Encryption Keys (CMEK)
- Egress Perimeter Controls
- Single Sign-On (SSO) for your CockroachDB Cloud organization and your clusters
- Network security



## Delete cluster

Proceed with the following steps only if you are sure you want to delete a cluster:

- Navigate to the Overview page for the cluster you want to delete.
- Click the Actions button in the top right corner.
- Select Delete cluster.
- In the confirmation window, enter the name of the cluster.
- Click Delete.

# Expanding Terraform Knowledge

# How to manage CockroachDB as Code with Terraform

## How to deploy CockroachDB Dedicated with Terraform

List of required prerequisites:

- Git CLI
- Terraform (=>1.3.1)
- Cockroach Cloud API Key

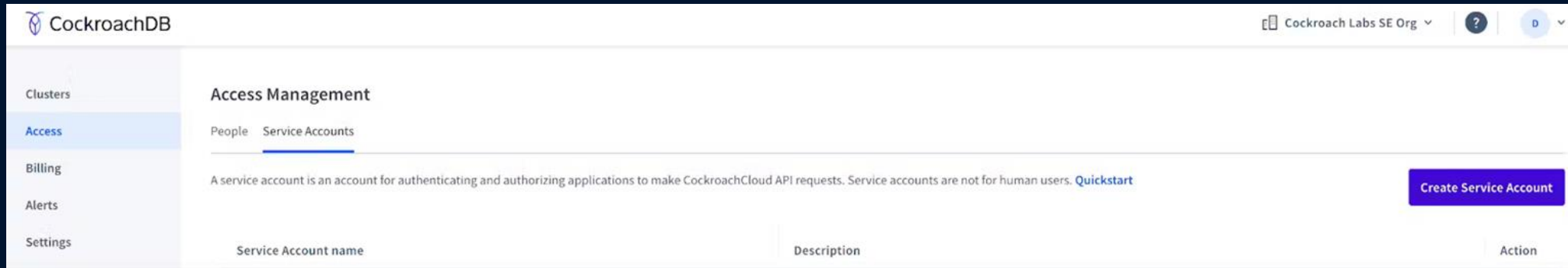


## Step 1: Clone the git repository

- git clone <https://github.com/cockroachlabs-field/cockroachdb-terraform-dedicated-july2023-example.git>

## Step 2: Grab your API key

Before we get into the Terraform code, and the resources it creates, we need to add the Cockroach API key as a local environment variable. You can create one of these in the Cockroach Cloud UI by adding a service account, which then allows you to generate a key associated with it.



Once a service account is created and an API key is generated we export that into our environment for Terraform to use.

- **export COCKROACH\_API\_KEY=<YOUR\_API\_KEY>**



### Step 3: Prepare your variables

- The new CockroachDB Cloud Terraform provider requires the presence of the environment variable to be able to create the required resources within the cloud platform. Now let's take a closer look at the Terraform code itself.

```
cluster_name = "dsmbcrdbtftexample2"
sql_user_name = "mbds"
sql_user_password = "ThisIsASuperSafePassword"
cloud_provider = "AWS"
cloud_provider_region = ["eu-west-2"]
cluster_nodes = "3"
storage_gib = "150"
machine_type = "m6i.xlarge"
region = "eu-west-2"
datadog_site = "US1"
datadog_api_key = "redacted"
```

## Step 4: Connection String and Certificate

- This is followed by the location of the certificate required to communicate with the cluster. There is an example below.

```
'postgresql://<user>@<cluster-name>-<short-id>.<region>.<host>:26257/<database>?sslmode=verify-full&sslrootcert='$HOME'/Library/CockroachCloud/certs/<cluster-name>-ca.crt'
```

- To retrieve this information with Terraform you need to create a data resource in your terraform code to retrieve it. Below is an example on how to do this.

```
data "cockroach_connection_string" "cockroach" {
 id = cockroach_cluster.cockroach.id
 sql_user = cockroach_sql_user.cockroach.name
 database = cockroach_cluster.cockroach.id
}
```

- With this, you can then create an output in your output.tf file to display the connection string when the terraform runs, or output to a variable to use elsewhere. Below is an example of outputting the connection string

```
output "connection_string" {
 value = module.cockroach-dedicated.connection_string
}
```

- Alongside the connection string, in order to connect to your cluster, you'll also need to output the cluster certificate, you can output this as a file within the terraform with the code snippet below.

```
resource "local_file" "cluster_cert" {
 filename = "${path.root}/cert.pem"
 content = data.cockroach_cluster_cert.cockroach.cert
}
```

- If required you can also output this as a variable or text output using the output snippet below:

```
output "cert" {
 value = data.cockroach_cluster_cert.cockroach.cert
}
```

## Step 5: Customer Managed Encryption Keys (CMEK)

Customer-Managed Encryption Keys (CMEK) allow you to protect data at rest in a CockroachDB Dedicated advanced private cluster using a cryptographic key that is entirely within your control, hosted in a supported cloud provider key-management system (KMS). This key is called the CMEK key.

You can manage your CMEK keys using one or more of the following services:

- Amazon Web Services (AWS) KMS
- Google Cloud Platform (GCP) KMS



## Step 6: Maintenance Windows

- Within Cockroach Cloud you can view and manage the patch upgrade window for your cluster.
- To help keep your clusters updated while minimizing disruption and downtime, set a window of time when your cluster is experiencing the lowest traffic.
- You are also able to configure this using Terraform.

```
resource "cockroach_maintenance_window" "example" {
 id = cockroach_cluster.cockroach.id
 offset_duration = var.offset_duration
 window_duration = var.window_duration
}
```

## Step 7: Operations with Datadog

- Cockroach Cloud allows for the export of metrics to external monitoring tools like Datadog.
- By doing this you can collate metrics for other sources to give you an end to end view of the service you are trying to deliver.
- This is critical in maintaining your prescribed SLA to your customers, whoever that may be.
- This could be external users of the platform or an internal system where the customers are internal members of staff. To configure and manage metrics export for your CockroachDB Dedicated cluster, use the metricexport endpoint for Datadog.
- Access to the metricexport endpoints requires a valid CockroachDB Cloud service account with the appropriate permissions (admin privilege or Cluster Admin role). To configure this using Terraform use the following code example to help.

```
resource "cockroach_metric_export_datadog_config" "example" {
 id = cockroach_cluster.cockroach.id
 site = var.datadog_site
 api_key = var.datadog_api_key
}
```

## Step 8: Initialize your Terraform

- To prepare our directory containing the terraform code we have just cloned from the GitHub repository we must run terraform init.
- By running this command it prepares our directory with all the required components that are defined in our code.
- For example downloading any Terraform providers or modules that are externally hosted. In this blog that will be the CockroachDB Cloud provider.

terraform init

## Step 9: Check your planned outcome

Now we are able to move to the next stage which is to run a terraform plan. The terraform plan command creates an execution plan, which lets you preview the changes that Terraform plans to make to your infrastructure. When Terraform creates a plan it:

- Reads the current state of any already-existing remote objects to make sure that the Terraform state is up-to-date.
- Compares the current configuration to the prior state and notes any differences.
- Proposes a set of change actions that should, if applied, make the remote objects match the configuration.

terraform plan

## Step 10: Deploy your infrastructure

- To build our cluster there is one final step to complete. Once we are happy with our terraform plan and the resources that it is going to create, then terraform apply can be executed.
- terraform apply -auto-approve
- This will create all the resources that are defined in the terraform code in the repository. It will now take a number of minutes as Terraform instructs the APIs of AWS and Cockroach Cloud.



# Security Fundamentals in CockroachDB

# CockroachDB Security Hardening

## Securing CockroachDB

- You're going to learn how to secure CockroachDB on three levels: network, transport and database:
- Network-level security makes your database accessible only by trusted nodes, securing it from the rest of the network.
- Transport-level security deals with network traffic encryption for the secure transportation of data.
- Database-level security defines the user's privileges towards the data in the database.

## Install CockroachDB

- When installing or upgrading CockroachDB, you'll need to install the latest version of CockroachDB. You can download the latest version of CockroachDB from CockroachDB's Releases Page.

## Network-level security for CockroachDB

### ➤ iptables firewall for CockroachDB

- You can utilize iptables when deploying your CockroachDB on Linux to limit which ports are open to TCP port 26257.

# Make sure not to drop established connections.

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

# Allow SSH.

```
iptables -A INPUT -p tcp -m state --state NEW --dport 22 -j ACCEPT
```

# For allowing it to listen to the specified port

```
iptables -A INPUT -p tcp -m state --state NEW -m tcp --dport 26257 -j ACCEPT
```

```
iptables -A INPUT -j REJECT --reject-with icmp-host-prohibited
```

# Allow all outbound, drop everything else inbound.

```
iptables -A OUTPUT -j ACCEPT
```

```
iptables -A INPUT -j DROP
```

```
iptables -A FORWARD -j DROP
```

# Make the rule persistent

```
sudo /usr/libexec/iptables/iptables.init save
```



- The CockroachDB rule above will allow anyone to connect to port 26257. You could make it more strict by only accepting connections from certain IP addresses or subnets:

# Only allow access to CockroachDB port from the local subnet.

```
iptables -A INPUT -p tcp -m state --state NEW --dport 26257 -s 192.168.1.0/24 -j ACCEPT
```

- As an example, following command with --listen-addr will configure CockroachDB to only listen on the localhost interface:

# For listening to port 26257 for traffic and 8080 for requests from admin UI

```
cockroach start --certs-dir=certs --store=node1 --listen-addr=localhost:26257 --http-addr=localhost:8080 --join=localhost:26257,localhost:26258,localhost:26259
```

- The procedure is as follows:

# Set up the --advertise-addr

```
cockroach start --certs-dir=certs --advertise-addr=localhost:26257 --join=localhost:26257,localhost:26258,localhost:26258 --cache=.25 --max-sql-memory=.25
```



## Network authorization

You can also set up authorization for your network based on the network type. You can utilize IP allowlisting, VPC peering and AWS PrivateLink:

- **IP allowlisting:** The process of allowing the connection to specific IP addresses by adding them to the allowlist of the CockroachDB Dedicated Cluster.
- **VPC peering:** This comes in handy if you're utilizing the Google Cloud Provider as it allows the CockroachDB cluster to only connect to internal addresses, keeping it safe from the public network.
- **AWS PrivateLink:** This can be utilized if you're using AWS as your cloud provider. It allows the secure connection between the AWS application and the CockroachDB cluster with the utilization of a private AWS endpoint. AWS endpoint is a URL that serves as an entry point for the web service.

- Follow the steps below to create an AWS endpoint:

# Step 1: Setting up AWS Endpoint

```
aws ec2 create-vpc-endpoint --region <region_name> --vpc-id <vpc_id> --subnet-ids
<Subnet1_ID> <Subnet2_ID> --vpc-endpoint-type Interface --security-group-ids
<SecurityGroup1_ID> <SecurityGroup2_ID> --service-name <Service_Name_Provided_By
_Cockroach>
```

# Step 2: Copy the Endpoint ID generated as output and verify it by pasting it into VPC Endpoint Field in Amazon VPC Console

# Step 3: Enable the DNS

```
aws ec2 modify-vpc-endpoint --region <Region_name> --private-dns-enabled --vpc-endpoint-id
<VPC_endpoint_ID>
```

# Step 4: After some delay, the status will be active, allowing you to connect

## Transport-level security for CockroachDB

- This section discusses the details of hardening CockroachDB on the transport level to protect it against security breaches. We'll be covering the details of cryptographic protocols—SSL/TLS and Client Certificate Authentication — for the secure and encrypted transportation of your data.
- This is what it looks like on a test CockroachDB Cloud database:

```
cockroach sql --url "postgresql:# p_1809:_RqtsBh1U4bBoL5BrIS4hg@free-tier8.aws-ap-southeast-1.cockroachlabs.cloud:26257/defaultdb?sslmode=verify-full&sslrootcert=certs/root.crt&options=--cluster%3Dtest-cluster-1594"
```

## Database-level security for CockroachDB

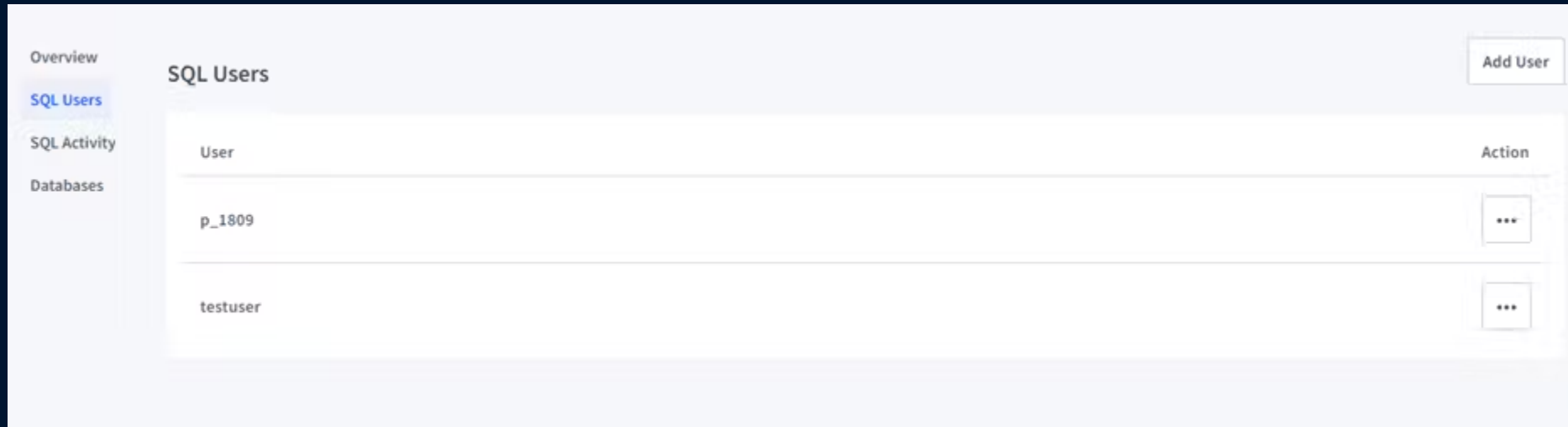
- This section covers the details of hardening your CockroachDB on the database level. It defines the set of privileges and actions for the user so that only the trusted users can access and alter the data. We'll be covering the details of user authentication, user authorization, audit logs, and encryption.

### User authorization

Authorization is the process of defining a user's privileges along with the set of actions they can perform. By default, the user holds all privileges. Permission can be altered using these steps:



- Check for the current users of your database under SQL Users.

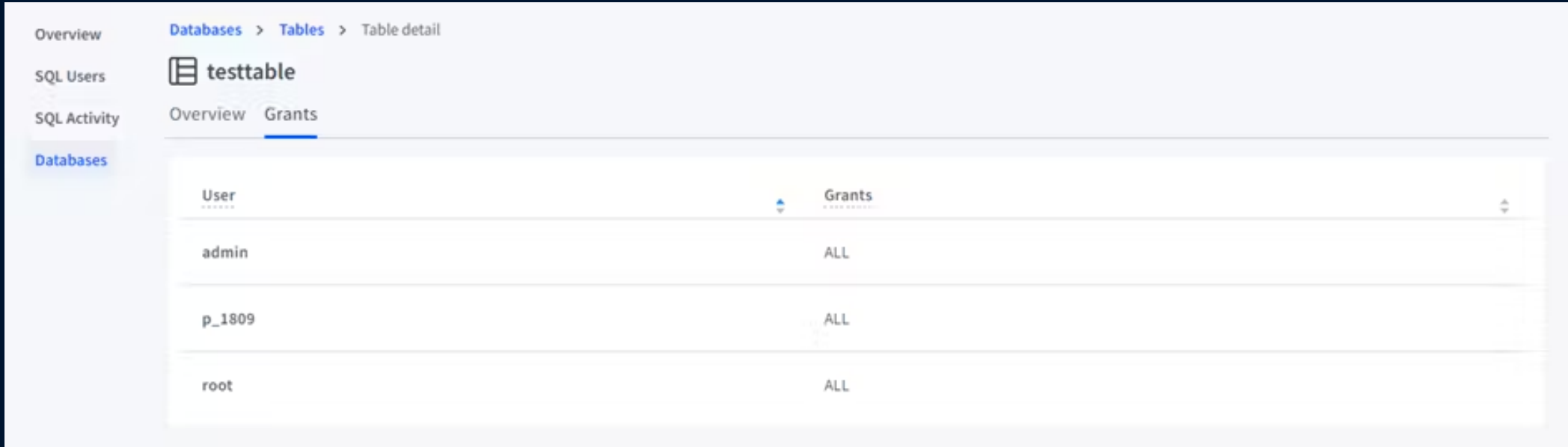


The screenshot shows a web interface for managing SQL users. On the left is a sidebar with navigation links: Overview, SQL Users (highlighted), SQL Activity, and Databases. The main area is titled 'SQL Users' and contains a table with two columns: 'User' and 'Action'. There is an 'Add User' button in the top right corner. The table lists two users: 'p\_1809' and 'testuser', each with a three-dot menu icon in the 'Action' column.

| User     | Action |
|----------|--------|
| p_1809   | ...    |
| testuser | ...    |

- or using SQL
- SHOW USERS;

- Look for their permissions under the Grants tab in Databases. By default, they'll have all permissions.



| Databases > Tables > Table detail |        |
|-----------------------------------|--------|
| testtable                         |        |
| Overview Grants                   |        |
| User                              | Grants |
| admin                             | ALL    |
| p_1809                            | ALL    |
| root                              | ALL    |

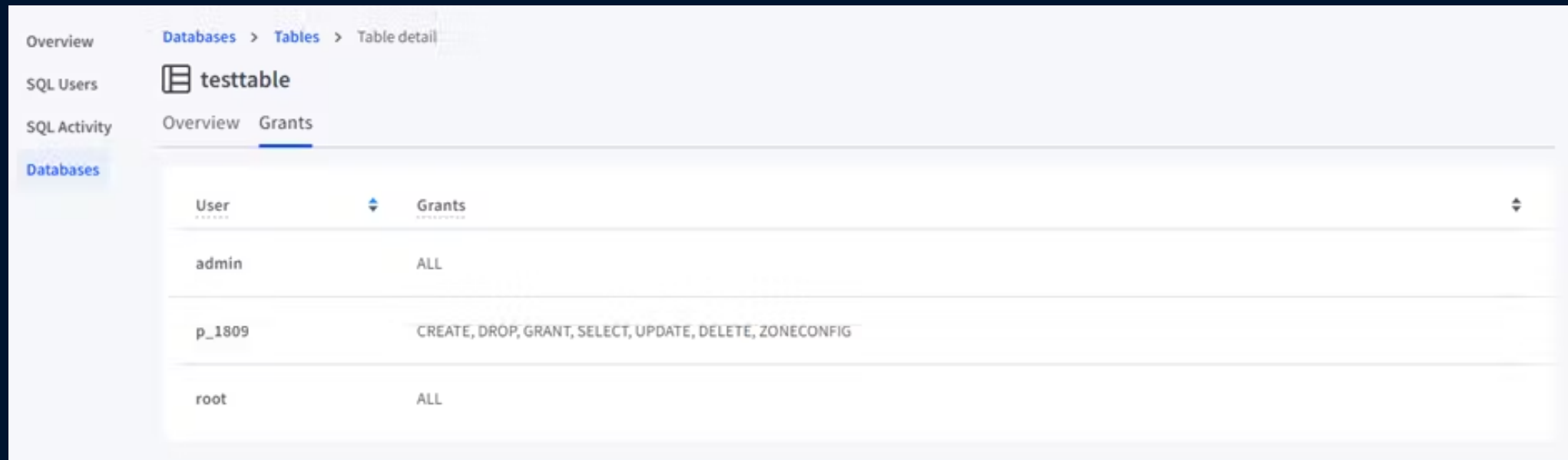
- or using SQL
- `SHOW GRANTS [ON [DATABASE | SCHEMA | TABLE | TYPE] <targets...>] [FOR <users...>]`

- Alter the permissions by revoking the conditions:

# Revoking privileges from the user

```
REVOKE INSERT ON TABLE <database>.<table> FROM <user>;
```

- Check the permissions after revoking to see if they've been updated.



The screenshot shows a web-based database management interface. On the left is a sidebar with navigation links: Overview, SQL Users, SQL Activity, and Databases (which is highlighted). The main content area has a breadcrumb trail: Databases > Tables > Table detail. Below this, there's a section for 'testtable' with two tabs: Overview and Grants (which is selected). The Grants tab displays a table with two columns: 'User' and 'Grants'. The table lists three users and their respective permissions.

| User   | Grants                                                  |
|--------|---------------------------------------------------------|
| admin  | ALL                                                     |
| p_1809 | CREATE, DROP, GRANT, SELECT, UPDATE, DELETE, ZONECONFIG |
| root   | ALL                                                     |

- If you want to grant permissions to the user, the command is as follows:

# Grant ALL permissions to the user on the table using db console.

```
GRANT ALL ON TABLE <db_name>.<table_name> TO <username>;
```

➤ Similarly, you can create user roles and assign them privileges using the following steps:

# Step 1: Create a user role

```
CREATE ROLE <name_of_role>;
```

# Step 2: Grant the privileges to the user

```
GRANT <privilege> ON DATABASE <name_of_database> TO <name_of_role>;
```



## Secure access to Self-Hosted CockroachDB using Teleport Database Access

# Step 1: Create export CockroachDB ca certificate, ca-key and client.root

```
tctl auth sign \
 --format=cockroachdb \
 --host=roach.example.com \
 --out=/path/to/cockroach/certs/dir/ \
 --ttl=2190h
```

# The command will produce 3 files: ca.crt with Teleport's certificate  
# authority and node.crt / node.key with the node's certificate and  
# key. Do not rename them as this is # how CockroachDB expects them  
# to be named.

# Step 2: Start CockroachDB with Teleport Cert

```
cockroach start \
 --certs-dir=/path/to/cockroachdb/certs/dir/ \
other flags...
```

# Step 3: Connect to CockroachDB using Teleport

# <https://goteleport.com/docs/database-access/guides/cockroachdb-self-hosted>

# The guide to secure deployments in CockroachDB

## Security

- CockroachDB provides two diametrically-opposed security modes governed by the flag `--insecure`. Let's see what happens when we use it:

```
$ cockroach start --insecure
```

```
*
```

```
* WARNING: RUNNING IN INSECURE MODE!
```

```
*
```

```
* - Your cluster is open for any client that can access <all your IP addresses>.
```

```
* - Any user, even root, can log in without providing a password.
```

```
* - Any user, connecting as root, can read or write any data in your cluster.
```

```
* - There is no network encryption nor authentication, and thus no confidentiality.
```

```
*
```

```
* Check out how to secure your cluster: https://www.cockroachlabs.com/docs/stable/secure-a-cluster
```

```
*
```

## Encryption

- Enabling secure mode changes all network communication to be done through TLS. A
- If traffic between CockroachDB nodes as well as client-server communications end up encrypted. You may start off thinking you have no need for in-flight encryption because your datacenter is secure and you have strict controls in place for which clients can even access your CockroachDB nodes.
- But before long you find yourself needing to add firewall rules for ad-hoc services running outside your carefully-controlled environment, or you decide to expand your CockroachDB cluster to another datacenter, communicating over untrusted network links.
- Without encryption, you now find yourself sending plaintext data over networks you are not in full control of. In this day and age we probably don't need to elaborate on the need for encrypted communication.

## Authentication

Authentication is the act of verifying the identity of the other party in a communication. In a secure CockroachDB deployment, we perform authentication in a few places:

- **CockroachDB nodes have the right client certificate**

All nodes should have a client certificate for the special user node. This is to restrict access to the node-to-node protocol.

- **Node addresses are in the server certificate**

The address (IP address or DNS name) used to reach a node, either directly or through a load balancer, should be listed in the server certificate presented by the node. This is needed to make sure we are indeed talking to a CockroachDB node, and not a man-in-the-middle.



## Monitoring & Alerting

- Another critical aspect of running anything in production (be it your database, monolithic server, or micro service) is monitoring. You should not be waiting for user complaints to know that your system is experiencing problems.
- CockroachDB comes with a built-in Admin UI showing you high-level metrics about various aspects of the system. While these are helpful in examining the current workload on your cluster, there is one thing it cannot do: alerting. The reason for this is quite simple: the worst of cases is when most or all of the cluster is down. Because the Admin UI is built into CockroachDB, it could not possibly alert you if your cluster were down.
- To solve this problem, we provide metrics in a format understood by Prometheus. Besides recording all metrics, Prometheus allows you to write rules for alerts to send to the AlertManager. In turn, AlertManager can notify you in any number of ways (email, slack, pagerduty, etc...) and make sure you are promptly notified of cockroach issues.
- We've hopefully convinced you of the need to run CockroachDB in secure mode, as well as setup monitoring. While a lot more goes into a good production deployment (provisioning, tooling, backups, etc.), security and monitoring are two of the early decisions you should not skip.

# Backup and Recovery in CockroachDB

# BACKUP

## Considerations

- Core users can only take full backups. To use the other backup features, you need an Enterprise license. You can also use CockroachDB Dedicated, which runs full backups daily and incremental backups hourly.
- Full cluster backups include Enterprise license keys. When you restore a full cluster backup that includes an Enterprise license, the Enterprise license is also restored.
- Zone configurations present on the destination cluster prior to a restore will be overwritten during a cluster restore with the zone configurations from the backed up cluster. If there were no customized zone configurations on the cluster when the backup was taken, then after the restore the destination cluster will use the zone configuration from the RANGE DEFAULT configuration.
- You cannot restore a backup of a multi-region database into a single-region database.
- Exclude a table's row data from a backup using the `exclude_data_from_backup` parameter.
- BACKUP is a blocking statement. To run a backup job asynchronously, use the DETACHED option. See the options below.



## ➤ Storage considerations

- HTTP storage is not supported for BACKUP and RESTORE.
- Modifying backup files in the storage location could invalidate a backup, and therefore, prevent a restore. In v22.1 and later, we recommend enabling object locking in your cloud storage bucket.
- While Cockroach Labs actively tests Amazon S3, Google Cloud Storage, and Azure Storage, we do not test S3-compatible services (e.g., MinIO, Red Hat Ceph).



## Required privileges

### ➤ Privileges for managing a backup job

To manage a backup job with PAUSE JOB, RESUME JOB, or CANCEL JOB, users must have at least one of the following:

- Be a member of the admin role.
- The CONTROLJOB role option.

To view a backup job with SHOW JOB, users must have at least one of the following:

- New in v23.1: The VIEWJOB privilege, which allows you to view all jobs (including admin-owned jobs).
- Be a member of the admin role.
- The CONTROLJOB role option.

See GRANT for detail on granting privileges to a role or user.

## Required privileges using the legacy privilege model

The following details the legacy privilege model that CockroachDB supports in v22.2 and earlier. Support for this privilege model will be removed in a future release of CockroachDB:

- Full cluster backups can only be run by members of the admin role. By default, the root user belongs to the admin role.
- For all other backups, the user must have read access on all objects being backed up. Database backups require CONNECT privileges, and table backups require SELECT privileges. Backups of user-defined schemas, or backups containing user-defined types, require USAGE privileges.

See the Required privileges section for the updated privilege model.

## Destination privileges

You can grant a user the EXTERNALIOIMPLICITACCESS system-level privilege.

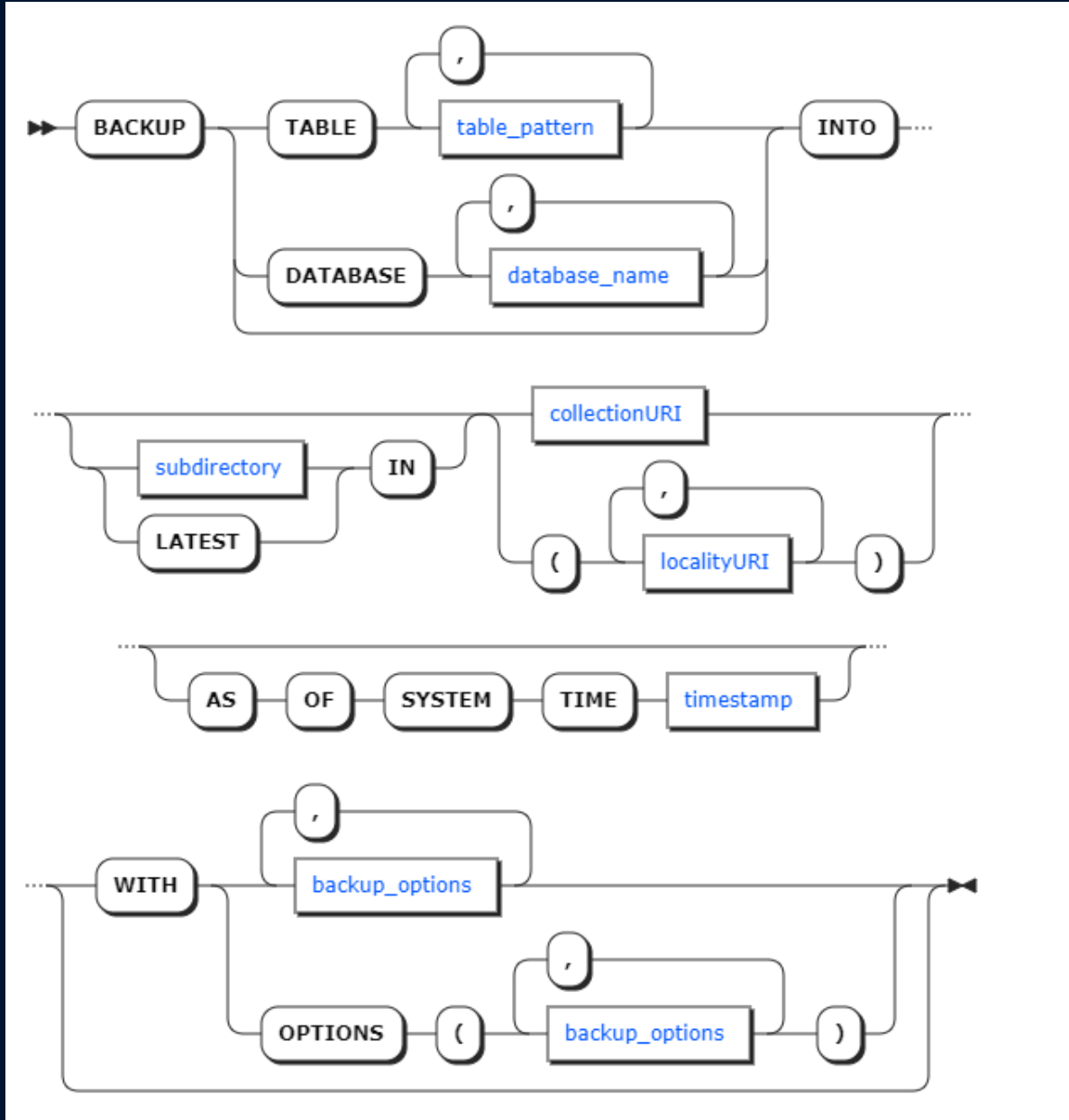
Either the EXTERNALIOIMPLICITACCESS system-level privilege or the admin role is required for the following scenarios:

- Interacting with a cloud storage resource using IMPLICIT authentication.
- Using a custom endpoint on S3.
- Using the cockroach nodelocal upload command.

No special privilege is required for:

- Interacting with an Amazon S3 and Google Cloud Storage resource using SPECIFIED credentials. Azure Storage is always SPECIFIED by default.
- Using Userfile storage.

## Synopsis





## Parameters

- Targets
- Query parameters
- Options

- Backup file URLs

CockroachDB uses the URL provided to construct a secure API call to the service you specify. The URL structure depends on the type of file storage you are using. For more information, see the following:

- URL format
- Example file URLs
- Authentication parameters

You can create an external connection to represent an external storage or sink URI. This allows you to specify the external connection's name in statements rather than the provider-specific URI. For detail on using external connections, see the [CREATE EXTERNAL CONNECTION](#) page.

## Functional details

### ➤ Object dependencies

Dependent objects must be backed up at the same time as the objects they depend on.

| Object                                    | Depends On                                                                           |
|-------------------------------------------|--------------------------------------------------------------------------------------|
| Table with <u>foreign key</u> constraints | The table it REFERENCES; however, this dependency can be removed during the restore. |
| Table with a <u>sequence</u>              | The sequence it uses; however, this dependency can be removed during the restore.    |
| Views                                     | The tables used in the view's SELECT statement.                                      |

### ➤ Users and privileges

The system.users table stores your users and their passwords. To restore your users and privilege grants, do a cluster backup and restore the cluster to a fresh cluster with no user data. You can also backup the system.users table, and then use this procedure.

## Performance

The backup job process minimizes its impact to the cluster's performance with:

- Distribution of work to all nodes. Each node backs up only a specific subset of the data it stores (those for which it serves writes), with no two nodes backing up the same data. Refer to the Backup Architecture page for a detailed explanation of how a backup job works.
- New in v23.1: Integration with elastic CPU limiter by default, which helps to minimize the impact backups have on foreground traffic. This integration will limit the amount of CPU time used by a backup thereby allowing foreground SQL traffic to continue largely unaffected.

### ➤ Backup performance configuration

- `bulkio.backup.file_size`
- `cloudstorage.azure.concurrent_upload_buffers`
- `cloudstorage.<provider>.write.node_rate_limit`
- `cloudstorage.<provider>.write.node_burst_limit`
- `cloudstorage.<provider>.read.node_rate_limit`
- `cloudstorage.<provider>.read.node_burst_limit`

## Viewing and controlling backups jobs

After CockroachDB successfully initiates a backup, it registers the backup as a job, and you can do the following:

| Action                 | SQL Statement |
|------------------------|---------------|
| View the backup status | SHOW JOBS     |
| Pause the backup       | PAUSE JOB     |
| Resume the backup      | RESUME JOB    |
| Cancel the backup      | CANCEL JOB    |

- You can also visit the Jobs page of the DB Console to view job details. The BACKUP statement will return when the backup is finished or if it encounters an error.



## Examples

Per our guidance in the Performance section, we recommend starting backups from a time at least 10 seconds in the past using AS OF SYSTEM TIME.

The examples in this section use one of the following storage URIs:

- External connections, which allow you to represent an external storage or sink URI. You can then specify the external connection's name in statements rather than the provider-specific URI. For detail on using external connections, see the CREATE EXTERNAL CONNECTION page.
- Amazon S3 connection strings with the default AUTH=specified parameter. For guidance on using AUTH=implicit authentication with Amazon S3 buckets instead, read Cloud Storage Authentication.

### ➤ Back up a cluster

- To take a full backup of a cluster:

```
BACKUP INTO 'external://backup_s3' AS OF SYSTEM TIME '-10s';
```

### ➤ **Back up a database**

- To take a full backup of a single database:

BACKUP DATABASE bank INTO 'external://backup\_s3' AS OF SYSTEM TIME '-10s';

- To take a full backup of multiple databases:

BACKUP DATABASE bank, employees INTO 'external://backup\_s3' AS OF SYSTEM TIME '-10s';

### ➤ **Back up a table or view**

- To take a full backup of a single table or view:

BACKUP bank.customers INTO 'external://backup\_s3' AS OF SYSTEM TIME '-10s';

- To take a full backup of multiple tables:

BACKUP bank.customers, bank.accounts INTO 'external://backup\_s3' AS OF SYSTEM TIME '-10s';

### ➤ **Back up all tables in a schema**

- To back up all tables in a schema, use a wildcard (\*) with the schema name:

BACKUP test\_schema.\* INTO 'external://backup\_s3' AS OF SYSTEM TIME '-10s';

## ➤ Create incremental backups

- When a BACKUP statement specifies an existing subdirectory in the collection, explicitly or via the LATEST keyword, an incremental backup will be added to the default /incrementals directory at the root of the collection storage location.
- To take an incremental backup using the LATEST keyword:  
BACKUP INTO LATEST IN 'external://backup\_s3' AS OF SYSTEM TIME '-10s';
- To store the backup in an existing subdirectory in the collection:  
BACKUP INTO {'subdirectory'} IN 'external://backup\_s3' AS OF SYSTEM TIME '-10s';

## ➤ Run a backup asynchronously

- Use the DETACHED option to execute the backup job asynchronously:  
BACKUP INTO 'external://backup\_s3' AS OF SYSTEM TIME '-10s' WITH DETACHED;
- The job ID is returned after the backup job creation completes:

```
job_id

592786066399264769
(1 row)
```

- Without the DETACHED option, BACKUP will block the SQL connection until the job completes. Once finished, the job status and more detailed job data is returned:

```
job_id | status | fraction_completed | rows | index_entries | bytes
-----+-----+-----+-----+-----+-----
652471804772712449 | succeeded | 1 | 50 | 0 | 4911
(1 row)
```



### ➤ Back up with an S3 storage class

- To associate your backup objects with a specific storage class in your Amazon S3 bucket, use the S3\_STORAGE\_CLASS parameter with the class. For example, the following S3 connection URI specifies the INTELLIGENT\_TIERING storage class:

```
BACKUP DATABASE movr INTO 's3://{BUCKET NAME}?AWS_ACCESS_KEY_ID={KEY
ID}&AWS_SECRET_ACCESS_KEY={SECRET ACCESS
KEY}&S3_STORAGE_CLASS=INTELLIGENT_TIERING' AS OF SYSTEM TIME '-10s';
```

- To use an external connection URI to back up to cloud storage with an associated S3 storage class, you need to include the S3\_STORAGE\_CLASS parameter when you create the external connection.
- Use the parameter to set one of these storage classes listed in Amazon's documentation. For more general usage information, see Amazon's Using Amazon S3 storage classes documentation.

## ➤ Advanced examples

For examples of advanced BACKUP and RESTORE use cases, see:

- Incremental backups with a specified destination
- Backup with revision history and point-in-time restore
- Locality-aware backup and restore
- Encrypted backup and restore
- Restore into a different database
- Remove the foreign key before restore
- Restoring users from system.users backup
- Show an incremental backup at a different location
- Exclude a table's data from backups

## See also

- Take Full and Incremental Backups
- Take and Restore Encrypted Backups
- Take and Restore Locality-aware Backups
- Take Backups with Revision History and Restore from a Point-in-time
- SHOW BACKUP
- CREATE SCHEDULE FOR BACKUP
- RESTORE
- Replication Controls

# RESTORE

## Considerations

- RESTORE cannot restore backups made by newer versions of CockroachDB.
- RESTORE only supports backups taken on a cluster on a specific major version into a cluster that is on the same version or the next major version. Refer to the Restoring Backups Across Versions page for more detail.
- RESTORE is a blocking statement. To run a restore job asynchronously, use the DETACHED option.
- RESTORE no longer requires an Enterprise license, regardless of the options passed to it or to the backup it is restoring.
- Zone configurations present on the destination cluster prior to a restore will be overwritten during a cluster restore with the zone configurations from the backed up cluster. If there were no customized zone configurations on the cluster when the backup was taken, then after the restore the destination cluster will use the zone configuration from the RANGE DEFAULT configuration.
- You cannot restore a backup of a multi-region database into a single-region database.
- When the `exclude_data_from_backup` parameter is set on a table, the table will not contain row data when restored.



## Required privileges

You can grant the RESTORE privilege to a user or role depending on the type of restore required:

| Restore  | Privilege                                                                                                                                                |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Cluster  | Grant a user the system level RESTORE privilege. For example, GRANT SYSTEM RESTORE TO user;.                                                             |
| Database | Grant a user the system level RESTORE privilege to restore databases onto the cluster. For example, GRANT SYSTEM RESTORE TO user;.                       |
| Table    | Grant a user the database level RESTORE privilege to restore schema objects into the database. For example, GRANT RESTORE ON DATABASE nonadmin TO user;. |

The listed privileges do not cascade to objects lower in the schema tree. For example, if you are granted system-level restore privileges, this does not give you the privilege to restore a table. If you need the RESTORE privilege on a database to apply to all newly created tables in that database, use DEFAULT PRIVILEGES. You can add RESTORE to the user or role's default privileges with ALTER DEFAULT PRIVILEGES.

## ➤ Privileges for managing a restore job

To manage a restore job with PAUSE JOB, RESUME JOB, or CANCEL JOB, users must have at least one of the following:

- Be a member of the admin role.
- The CONTROLJOB role option.

To view a restore job with SHOW JOB, users must have at least one of the following:

- New in v23.1: The VIEWJOB privilege, which allows you to view all jobs (including admin-owned jobs).
- Be a member of the admin role.
- The CONTROLJOB role option.

See GRANT for detail on granting privileges to a role or user.

## Required privileges using the legacy privilege model

The following details the existing privilege model that CockroachDB supports in v22.2 and earlier. Support for this privilege model will be removed in a future release of CockroachDB:

- Full cluster restores can only be run by members of the ADMIN role. By default, the root user belongs to the admin role.
- For all other restores, the user must have write access (CREATE or INSERT) on all objects affected.

See the Required privileges section for the updated privilege model.

## Source privileges

You can grant a user the EXTERNALIOIMPLICITACCESS system-level privilege to interact with external resources that require implicit access.

Either the EXTERNALIOIMPLICITACCESS system-level privilege or the admin role is required for the following scenarios:

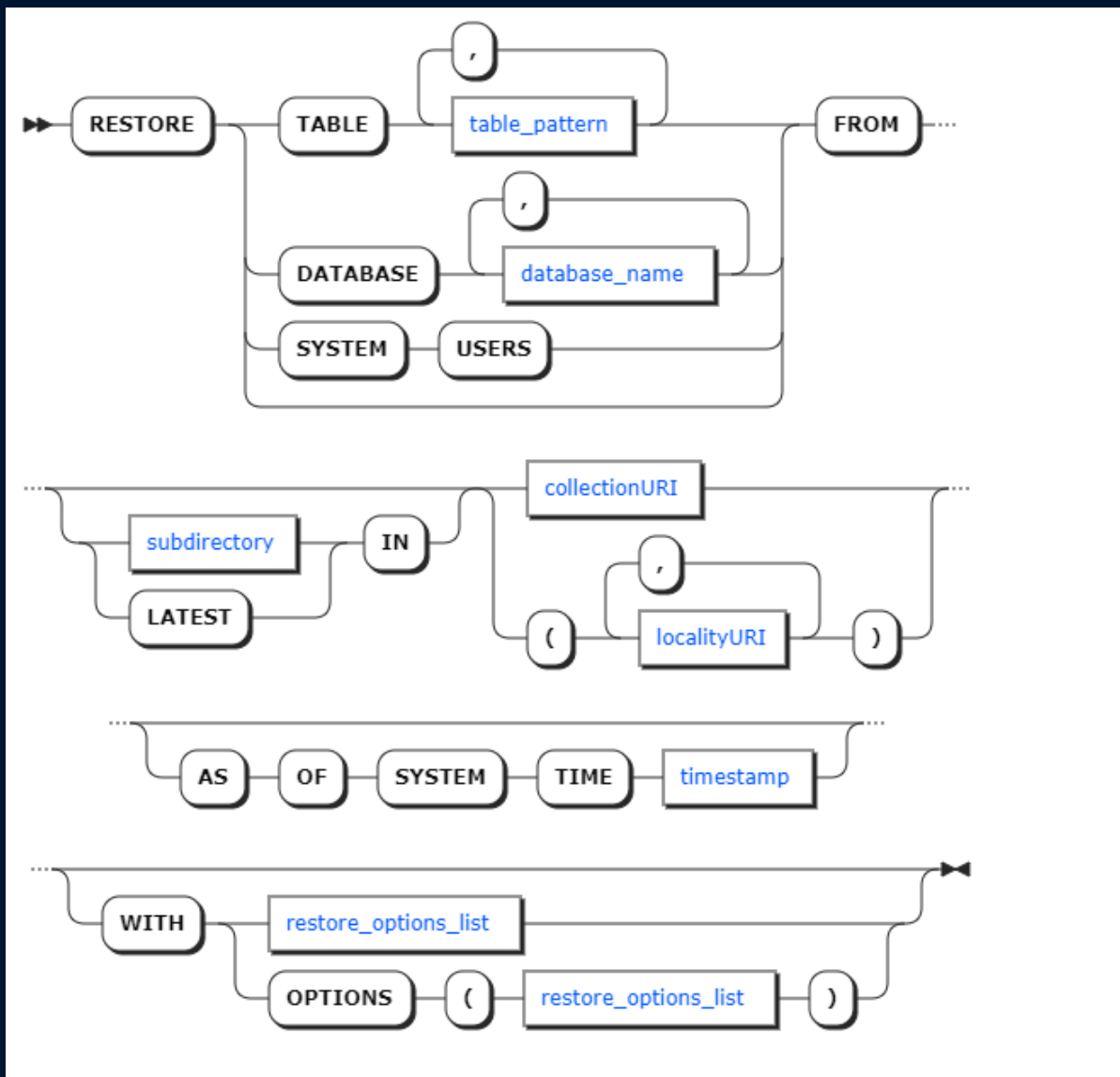
- To interact with a cloud storage resource using IMPLICIT authentication.
- Use of a custom endpoint on S3.
- Nodelocal

No special privilege is required for:

- Interacting with an Amazon S3 and Google Cloud Storage resource using SPECIFIED credentials. Azure Storage is always SPECIFIED by default.
- Using Userfile storage.



## Synopsis



## Parameters

### ➤ Options

#### ➤ Backup file URLs

CockroachDB uses the URL provided to construct a secure API call to the service you specify. The URL structure depends on the type of file storage you are using. For more information, see the following:

- Use Cloud Storage
- Use a Local File Server

You can create an external connection to represent an external storage or sink URI. This allows you to specify the external connection's name in statements rather than the provider-specific URI. For detail on using external connections, see the [CREATE EXTERNAL CONNECTION](#) page.

## Functional details

You can restore:

- A full cluster
- Databases
- Tables

RESTORE will only restore the latest data in an object (table, database, cluster), or the latest data as per an AS OF SYSTEM TIME restore. That is, a restore will not include historical data even if you ran your backup with revision\_history.

This means that if you issue an AS OF SYSTEM TIME query on a restored object, the query will fail or the response will be incorrect because there is no historical data to query. For example, if you restore a table at 2022-07-13 10:38:00, it is not then possible to read or back up that table at 2022-07-13 10:37:00 or earlier. This is also the case for backups with revision\_history that might try to initiate a revision start time earlier than 2022-07-13 10:38:00.

## ➤ Full cluster

A full cluster restore can only be run on a target cluster with no user-created databases or tables. Restoring a full cluster includes:

- All user tables
- Relevant system tables
- All databases
- All tables (which automatically includes their indexes)
- All views

Also, consider that:

- Temporary tables will be restored to their original database during a full cluster restore.
- The restore will drop the cluster's defaultdb and postgres pre-loaded databases before the restore begins. You can only restore defaultdb and postgres if they are present in the original backup.
- Changefeed jobs will not resume automatically on the new cluster. It is necessary to manually create changefeeds after a full-cluster restore.
- When the cluster is in a mixed-version state during an upgrade, a full cluster restore will fail. To perform a full cluster restore, it is necessary to first finalize the upgrade.



## ➤ Databases

- Restoring a database will create a new database and restore all of its tables and views. The created database will have the name of the database in the backup.
- `RESTORE DATABASE backup_database_name FROM LATEST in 'your_backup_collection_URI';`

## ➤ Tables

- The target database must not have tables or views with the same name as the tables or views you're restoring. If any of the restore target's names are being used, you can:
- `DROP TABLE`, `DROP VIEW`, or `DROP SEQUENCE` and then restore them. Note that a sequence cannot be dropped while it is being used in a column's `DEFAULT` expression, so those expressions must be dropped before the sequence is dropped, and recreated after the sequence is recreated. The `setval` function can be used to set the value of the sequence to what it was previously.
- Restore the table or view into a different database.

When restoring an individual table that references a user-defined type (e.g., ENUM), CockroachDB will first check to see if the type already exists. The restore will attempt the following for each user-defined type within a table backup:

- If there is not an existing type in the cluster with the same name, CockroachDB will create the user-defined type as it exists in the backup.
- If there is an existing type in the cluster with the same name that is compatible with the type in the backup, CockroachDB will map the type in the backup to the type in the cluster.
- If there is an existing type in the cluster with the same name but it is not compatible with the type in the backup, the restore will not succeed and you will be asked to resolve the naming conflict. You can do this by either dropping or renaming the existing user-defined type.

In general, two types are compatible if they are the same kind (e.g., an enum is only compatible with other enums). Additionally, enums are only compatible if they have the same ordered set of elements that have also been created in the same way. For example:

- `CREATE TYPE t1 AS ENUM ('yes', 'no')` and `CREATE TYPE t2 AS ENUM ('yes', 'no')` are compatible.
- `CREATE TYPE t1 AS ENUM ('yes', 'no')` and `CREATE TYPE t2 AS ENUM ('no', 'yes')` are not compatible.
- `CREATE TYPE t1 AS ENUM ('yes', 'no')` and `CREATE TYPE t2 AS ENUM ('yes');` `ALTER TYPE t2 ADD VALUE ('no');` are not compatible because they were not created in the same way.

## ➤ Object dependencies

Dependent objects must be restored at the same time as the objects they depend on.

| Object                             | Depend On                                                                             |
|------------------------------------|---------------------------------------------------------------------------------------|
| Table with foreign key constraints | The table it REFERENCES (however, this dependency can be removed during the restore). |
| Table with a sequence              | The sequence.                                                                         |
| Views                              | The tables used in the view's SELECT statement.                                       |

Referenced UDFs are not restored and require the `skip_missing_udfs` option.

## ➤ Users and privileges

The owner of restored objects will be the user running the restore job. To restore your users and privilege grants, you can do a cluster backup and restore the cluster to a fresh cluster with no user data.

If you are not doing a full cluster restore, the table-level privileges need to be granted to the users after the restore is complete. (By default, the user restoring will become the owner of the restored objects.) To grant table-level privileges after a restore, backup the `system.users` table, restore users and their passwords, and then grant the table-level privileges.

## ➤ Restore types

You can either restore from a full backup or from a full backup with incremental backups, based on the backup files you include:

| Restore Type                      | Parameters                                                                                                                                                        |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Full backup                       | Include the path to the full backup destination and the subdirectory of the backup. See the Examples section for syntax of cluster, database, and table restores. |
| Full backup + incremental backups | Include the path that contains the backup collection and the subdirectory containing the incremental backup. See Restore from incremental backups for an example. |



## Performance

- The RESTORE process minimizes its impact to the cluster's performance by distributing work to all nodes. Subsets of the restored data (known as ranges) are evenly distributed among randomly selected nodes, with each range initially restored to only one node. Once the range is restored, the node begins replicating it others.
- When a RESTORE fails or is canceled, partially restored data is properly cleaned up. This can have a minor, temporary impact on cluster performance.
- A restore job will pause if a node in the cluster runs out of disk space. See Viewing and controlling restore jobs for information on resuming and showing the progress of restore jobs.
- A restore job will pause instead of entering a failed state if it continues to encounter transient errors once it has retried a maximum number of times. Once the restore has paused, you can either resume or cancel it.

## Restoring to multi-region databases

Restoring to a multi-region database is supported with some limitations. This section outlines details and settings that should be considered when restoring into multi-region databases:

- A cluster's regions will be checked before a restore. Mismatched regions between backup and restore clusters will be flagged before the restore begins, which allows for a decision between updating the cluster localities or restoring with the `skip_localities_check` option to continue with the restore regardless.
- A database that is restored with the `sql.defaults.primary_region` cluster setting will have the PRIMARY REGION from this cluster setting assigned to the target database.
- RESTORE supports restoring non-multi-region tables into a multi-region database and sets the table locality as REGIONAL BY TABLE to the primary region of the target database.

- Restoring tables from multi-region databases with table localities set to REGIONAL BY ROW, REGIONAL BY TABLE, REGIONAL BY TABLE IN PRIMARY REGION, and GLOBAL to another multi-region database is supported.
- When restoring a REGIONAL BY TABLE IN PRIMARY REGION table, if the primary region is different in the source database to the target database this will be implicitly changed on restore.
- Restoring a partition of a REGIONAL BY ROW table is not supported.
- REGIONAL BY TABLE and REGIONAL BY ROW tables can be restored only if the regions of the backed-up table match those of the target database. All of the following must be true for RESTORE to be successful:
  - The regions of the source database and the regions of the destination database have the same set of regions.
  - The regions were added to each of the databases in the same order.
  - The databases have the same primary region.

## Viewing and controlling restore jobs

- After CockroachDB successfully initiates a restore, it registers the restore as a job, which you can view with SHOW JOBS.
- After the restore has been initiated, you can control it with PAUSE JOB, RESUME JOB, and CANCEL JOB.

## Examples

There are two ways to specify a full or incremental backup to restore:

- Restoring from the most recent backup
- Restoring from a specific backup



## ➤ View the backup subdirectories

- BACKUP ... INTO adds a backup to a backup collection location. To view the backup paths in a given collection location, use SHOW BACKUPS:

```
SHOW BACKUPS IN 's3://bucket/path?AUTH=implicit';
path
```

```

/2023/12/14-190909.83
/2023/12/20-155249.37
/2023/12/21-142943.73
(3 rows)
```

## ➤ Restore the most recent full or incremental backup

- To restore from the most recent backup (full or incremental) in the collection's location, use the LATEST syntax:

```
RESTORE FROM LATEST IN 's3://bucket/path?AUTH=implicit';
```

## ➤ Restore a specific full or incremental backup

- To restore a specific full or incremental backup, specify that backup's subdirectory in the RESTORE statement.
- To view the available subdirectories, use SHOW BACKUPS. If you are restoring an incremental backup, the URI must point to the storage location that contains the full backup:

```
RESTORE FROM '2023/03/23-213101.37' IN 's3://bucket/path?AUTH=implicit';
```

## ➤ Restore a cluster

- To restore a full cluster:

```
RESTORE FROM LATEST IN 'external://backup_s3';
```

To view the available subdirectories, use SHOW BACKUPS.

## ➤ Restore a database

- To restore a database:

```
RESTORE DATABASE bank FROM LATEST IN 'external://backup_s3';
```

To view the available subdirectories, use SHOW BACKUPS.

## ➤ Restore a table

- To restore a single table:

```
RESTORE TABLE bank.customers FROM LATEST IN 'external://backup_s3';
```

- To restore multiple tables:

```
RESTORE TABLE bank.customers, bank.accounts FROM LATEST IN 'external://backup_s3';
```

To view the available subdirectories, use SHOW BACKUPS.

## ➤ Restore with AS OF SYSTEM TIME

## ➤ Restore a backup asynchronously

- Use the DETACHED option to execute the restore job asynchronously:

```
RESTORE FROM LATEST IN 'external://backup_s3'
WITH DETACHED;
```

- The job ID is returned after the restore job creation completes:

```
job_id

592786066399264769
(1 row)
```

- Without the DETACHED option, RESTORE will block the SQL connection until the job completes. Once finished, the job status and more detailed job data is returned:

```
job_id | status | fraction_completed | rows | index_entries | bytes
-----+-----+-----+-----+-----+-----
652471804772712449 | succeeded | 1 | 50 | 0 | 4911
(1 row)
```



## ➤ Other restore usages

- Restore tables into a different database
- By default, tables and views are restored to the database they originally belonged to. However, using the `into_db` option, you can control the target database. Note that the target database must exist prior to the restore.
- Create the new database that you'll restore the table or view into:  
`CREATE DATABASE newdb;`
- Restore the table into the newly created database with `into_db`:  
`RESTORE bank.customers FROM LATEST IN 'external://backup_s3'`  
`WITH into_db = 'newdb';`

## Known limitations

- To successfully restore a table into a multi-region database, it is necessary for the order and regions to match between the source and destination database.
- See the Known Limitations page for detail on ordering and matching regions. Tracking GitHub Issue
- Restoring GLOBAL and REGIONAL BY TABLE tables into a non-multi-region database is not supported. Tracking GitHub Issue

## See also

- BACKUP
- Take Full and Incremental Backups
- Take and Restore Encrypted Backups
- Take and Restore Locality-aware Backups
- Take Backups with Revision History and Restore from a Point-in-time
- Manage a Backup Schedule
- Replication Controls
- ENUM
- CREATE TYPE
- DROP TYPE



# Simple Monitoring and Troubleshooting



# Monitoring and Alerting

## Built-in monitoring tools

- CockroachDB includes several tools to help you monitor your cluster's workloads and performance.

### ➤ DB Console

- The DB Console collects time-series cluster metrics and displays basic information about a cluster's health, such as node status, number of unavailable ranges, and queries per second and service latency across the cluster.
- This tool is designed to help you optimize cluster performance and troubleshoot issues. The DB Console is accessible from every node at `http://<host>:<http-port>`, or `http://<host>:8080` by default.
- The DB Console automatically runs in the cluster. The following sections describe some of the pages that can help you to monitor and observe your cluster. For more information on accessing the DB Console, see Access DB Console.

## ➤ **Cluster API**

- The Cluster API is a REST API that runs in the cluster and provides much of the same information about your cluster and nodes as is available from the DB Console or the Prometheus endpoint, and is accessible from each node at the same address and port as the DB Console.
- If the cluster is unavailable, the Cluster API is also unavailable.
- For more information, see the Cluster API overview and reference.

## ➤ **crdb\_internal system catalog**

- The crdb\_internal system catalog is a schema in each database that contains information about internal objects, processes, and metrics about that database.
- DBMarlin provides a third-party tool that collects metrics from a cluster's crdb\_internal system catalogs rather than the cluster's Prometheus endpoint. Refer to Monitor CockroachDB with DBmarlin.
- If the cluster is unavailable, a database's crdb\_internal system catalog cannot be queried.
- For details, see crdb\_internal.

## ➤ Health endpoints

- CockroachDB provides two HTTP endpoints for checking the health of individual nodes.
- These endpoints are also available through the Cluster API under `/v2/health/`.
- If the cluster is unavailable, these endpoints are also unavailable.

## ➤ Raw status endpoints

- Several endpoints return raw status metrics in JSON at `http://<host>:<http-port>/#/debug`. Feel free to investigate and use these endpoints, but note that they are subject to change.

### Raw Status Endpoints (JSON)

|                         |                                              |                                                       |
|-------------------------|----------------------------------------------|-------------------------------------------------------|
| Logs (single node only) | On a Specific Node                           | <code>/_status/logs/[node_id]</code>                  |
|                         | Log Files                                    | <code>/_status/logfiles/[node_id]</code>              |
|                         | Specific Log File                            | <code>/_status/logfiles/[node_id]/[filename]</code>   |
| Metrics                 | Variables                                    | <code>/debug/metrics</code>                           |
|                         | Prometheus                                   | <code>/_status/vars</code>                            |
| Node Status             | All Nodes                                    | <code>/_status/nodes</code>                           |
|                         | Single node status                           | <code>/_status/nodes/[node_id]</code>                 |
| Single Node Specific    | Gossip                                       | <code>/_status/gossip/[node_id]</code>                |
|                         | Ranges                                       | <code>/_status/ranges/[node_id]</code>                |
|                         | Stacks                                       | <code>/_status/stacks/[node_id]</code>                |
|                         | Certificates                                 | <code>/_status/certificates/[node_id]</code>          |
| Sessions                | Local Sessions                               | <code>/_status/local_sessions</code>                  |
|                         | All Sessions                                 | <code>/_status/sessions</code>                        |
| Cluster Wide            | Raft                                         | <code>/_status/raft</code>                            |
|                         | Range                                        | <code>/_status/range/[range_id]</code>                |
|                         | Range Log                                    | <code>/_admin/v1/rangelog?limit=100</code>            |
|                         | Range Log for Specific Range                 | <code>/_admin/v1/rangelog/[range_id]?limit=100</code> |
| Allocator               | Simulated Allocator Runs on a Specific Node  | <code>/_status/allocator/node/[node_id]</code>        |
|                         | Simulated Allocator Runs on a Specific Range | <code>/_status/allocator/range/[range_id]</code>      |

## ➤ **Node status command**

The cockroach node status command gives you metrics about the health and status of each node.

- With the --ranges flag, you get granular range and replica details, including unavailability and under-replication.
- With the --stats flag, you get granular disk usage details.
- With the --decommission flag, you get details about the node decommissioning process.
- With the --all flag, you get all of the above.

## ➤ **Prometheus endpoint**

- Every node of a CockroachDB cluster exports granular time-series metrics at `http://<host>:<http-port>/_status/vars`. The metrics are formatted for easy integration with Prometheus, an open source tool for storing, aggregating, and querying time-series data.
- The Prometheus format is human-readable and can be processed to work with other third-party monitoring systems such as Sysdig and stackdriver.
- Many of the third-party monitoring integrations, such as Datadog and Kibana, collect metrics from a cluster's Prometheus endpoint.



## ➤ Critical nodes endpoint

The critical nodes status endpoint is used to:

- Check if any of your nodes are in a critical state. A node is critical if that node becoming unreachable would cause replicas to become unavailable.
- Check if any ranges are under-replicated or unavailable. This is useful when determining whether a node is ready for decommissioning.
- Check if any of your cluster's data placement constraints (set via multi-region SQL or direct configuration of replication zones) are being violated. This is useful when implementing data domiciling.

## Alerting tools

- In addition to actively monitoring the overall health and performance of a cluster, it is also essential to configure alerting rules that promptly send notifications when CockroachDB experiences events that require investigation or intervention.
- Many of the third-party monitoring integrations, such as Datadog and Kibana, also support event-based alerting using metrics collected from a cluster's Prometheus endpoint. Refer to the documentation for an integration for more details.
- This section identifies the most important events that you might want to create alerting rules for, and provides pre-defined rules definitions for these events appropriate for use with Prometheus's open source Alertmanager service.

## ➤ Alertmanager

- If you have configured Prometheus to monitor your CockroachDB instance, you can also configure alerting rule definitions to have Alertmanager detect important events and alert you when they occur.
- If you rely on external tools for storing and visualizing your cluster's time-series metrics, Cockroach Labs recommends that you disable the DB Console's storage of time-series metrics.
- When storage of time-series metrics is disabled, the DB Console Metrics dashboards in the DB Console are still available, but their visualizations are blank. This is because the dashboards rely on data that is no longer available.

## ➤ Events to alert on

- Node is down
- Node is restarting too frequently
- Node is running low on disk space
- Node is not executing SQL
- CA certificate expires soon
- Changefeed is experiencing high latency
- Unavailable ranges
- Tripped replica circuit breakers
- Under-replicated ranges
- Requests stuck in Raft
- High open file descriptor count



## See also

- [Production Checklist](#)
- [Manual Deployment](#)
- [Orchestrated Deployment](#)
- [Local Deployment](#)
- [Third-Party Monitoring Integrations](#)
- [Metrics](#)

# Common Errors and Solutions

## connection refused

This message indicates a client is trying to connect to a node that is either not running or is not listening on the specified interfaces (i.e., hostname or port).

To resolve this issue, do one of the following:

- If the node hasn't yet been started, start the node.
- If you specified a `--listen-addr` and/or a `--advertise-addr` flag when starting the node, you must include the specified IP address/hostname and port with all other cockroach commands or change the `COCKROACH_HOST` environment variable.

If you're not sure what the IP address/hostname and port values might have been, you can look in the node's logs.

If necessary, you can also shut down and then restart the node:

- `cockroach start [flags]`

### **node is running secure mode, SSL connection required**

- This message indicates that the cluster is using TLS encryption to protect network communication, and the client is trying to open a connection without using the required TLS certificates.
- To resolve this issue, use the `cockroach cert create-client` command to generate a client certificate and key for the user trying to connect. For a secure deployment tutorial, including generating security certificates and connecting clients, see Manual Deployment.

### **restart transaction**

Messages with the error code 40001 and the string restart transaction are known as transaction retry errors. These indicate that a transaction failed due to contention with another concurrent or recent transaction attempting to write to the same data. The transaction needs to be retried by the client.

In most cases, the correct actions to take when encountering transaction retry errors are:

- Update your application to support client-side retry handling when transaction retry errors are encountered. Follow the guidance for the specific error type.
- Take steps to minimize transaction retry errors in the first place. This means reducing transaction contention overall, and increasing the likelihood that CockroachDB can automatically retry a failed transaction.

**node belongs to cluster <cluster ID> but is attempting to connect to a gossip network for cluster <another cluster ID>**

This message usually indicates that a node tried to connect to a cluster, but the node is already a member of a different cluster. This is determined by metadata in the node's data directory. To resolve this issue, do one of the following:

- Choose a different directory to store the CockroachDB data:  
`cockroach start [flags] --store=[new directory] --join=[cluster host]:26257`
- Remove the existing directory and start a node joining the cluster again:  
`rm -r cockroach-data/  
cockroach start [flags] --join=[cluster host]:26257`



**open file descriptor limit of <number> is under the minimum required <number>**

- CockroachDB can use a large number of open file descriptors, often more than is available by default.
- This message indicates that the machine on which a CockroachDB node is running is under CockroachDB's recommended limits.
- For more details on CockroachDB's file descriptor limits and instructions on increasing the limit on various platforms, see [File Descriptors Limit](#).

## split failed while applying backpressure; are rows updated in a tight loop?

One possible cause is that the range consists only of MVCC version data due to a row being repeatedly updated, and the range cannot be split because doing so would spread MVCC versions for a single row across multiple ranges.

To resolve this issue, make sure you are not repeatedly updating a single row. If frequent updates of a row are necessary, consider one of the following:

- Reduce the gc.ttlseconds variable in the applicable zone configuration to reduce the garbage collection period and prevent such a large build-up of historical values.
- If a row contains large columns that are not being updated with other columns, put the large columns in separate column families.

## context deadline exceeded

- This message occurs when a component of CockroachDB gives up because it was relying on another component that has not behaved as expected, for example, another node dropped a network connection.
- To investigate further, look in the node's logs for the primary failure that is the root cause.

## protected ts verification error

To resolve this issue, take a new full backup after doing either of the following:

- Increase the garbage collection period by configuring the `gc.ttlseconds` replication zone variable. For example, we recommend setting the GC TTL to a time interval greater than the sum of `incremental_backup_interval` + `expected_runtime_of_full_backup` + `buffer_for_slowdowns`.
- To estimate the expected full backup runtime, it is necessary to perform testing or verify the past performance through the jobs table.

Increase the frequency of incremental backups.

## result is ambiguous

- In a distributed system, some errors can have ambiguous results. For example, if you receive a connection closed error while processing a COMMIT statement, you cannot tell whether the transaction successfully committed or not.
- These errors are possible in any database, but CockroachDB is somewhat more likely to produce them than other databases because ambiguous results can be caused by failures between the nodes of a cluster.
- These errors are reported with the PostgreSQL error code 40003 (statement\_completion\_unknown) and the message result is ambiguous.

## checking for key collisions: ingested key collides with an existing one

- When importing into an existing table with IMPORT INTO, this error occurs because the rows in the import file conflict with an existing primary key or another UNIQUE constraint on the table.
- The import will fail as a result. IMPORT INTO is an insert-only statement, so you cannot use it to update existing rows. To update rows in an existing table, use the UPDATE statement.



## memory budget exceeded

- If a node runs out of its allocated SQL memory (the memory allocated to the SQL layer), a memory budget exceeded error occurs.
- To mitigate this issue, ensure that the node has enough RAM and that enough memory is allocated to the SQL layer. The best approach depends upon the cluster's workload. Try the following approaches:
- If you find queries that are consuming too much memory, cancel the queries to free up memory usage. For information on optimizing query performance, see SQL Performance Best Practices.
- Increase the amount of memory on the node. Cockroach Labs recommends that you use the same hardware, operating system, and software configuration on each node.
- Increase `--max-sql-memory` on the node. A memory budget exceeded error is an early warning that the cockroach process on a node is at risk of crashing due to an out-of-memory (OOM) crash. To protect the node, CockroachDB fails the query.

# Thank you