DEVSISTERS

How Devsisters Launches Blockbuster Games in New Countries with Ease

ChangWon Lee, Lead DevOps Engineer, Devsisters

RoachFest24

About Devsisters and the CookieRun Franchise

- **Devsisters** is a South Korea-based mobile game developing company
- Develops and publishes the **CookieRun** franchise mobile games
- The **CookieRun** franchise has over 200 million users worldwide

Pre-register Now!

CookieRun (2013)

Side-scrolling running action platformer

CookieRun: Kingdom (2021)

Collecting RPG & city-building

CookieRun: Witch's Castle (2024)

Puzzle blockbuster adventure

CookieRun: Tower of Adventure

Casual multiplayer co-op action RPG

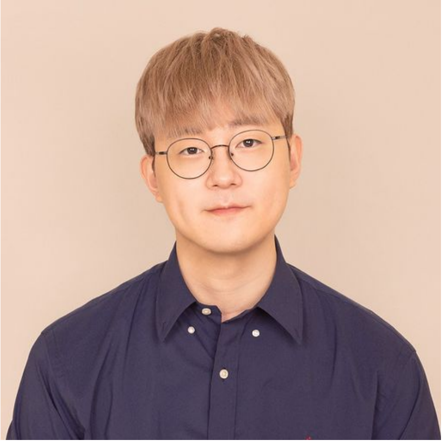
ChangWon Lee

- Lead DevOps Engineer @ Devsisters

Pre-register Now!

About My Team

- A central DevOps team within the Publishing Platform Group
  - Oversees DevOps, SRE, infrastructure provisioning, and platform engineering
  - Provides software architecture consultation to address complex requirements for new game launches
  - Implements abstraction layers and guardrails for fail-safe infrastructure provisioning



Being A Central DevOps Team Means…

Challenges Faced

- Game development teams bring unique and varied needs
- Directly fulfilling each requirements is practically impossible
  - 100–250 engineers vs. ~10 DevOps engineers

Our Approach

- Consolidate requirements into one robust solution
- Building blocks:
  - Cloud Infrastructure
  - Kubernetes
  - Helm Chart

CookieRun franchise powered by CockroachDB

CookieRun: Kingdom

- First title to use CRDB
- Successfully launched in **China** with **Changyou and Tencent** as publishing partner

CookieRun

- Migrating database from Couchbase
- Planning to launch in **India** with **Krafton** as publishing partner

CockroachDB

CookieRun: Witch's Castle

- Puzzle genre #1 in KR, TW, HK, SG, TH

CookieRun: Tower of Adventures

- **Coming soon on June 25!**
- Planning to launch in **Japan** with **Yostar** as publishing partner

Contents Flow

01 Introduction

Devsisters and CookieRun
Our Team’s Mission

02 Why we use CockroachDB

Rationale Behind Technical Decision

03 Operating CockroachDB

Tips from our experience

04 Deploying CockroachDB

Concept of “Serverfarms”

05 Conclusion

Quick Wrap Up

Why we use CockroachDB

CASE STUDY

DEVSISTERS

Sweet success: The global developer gaming platform

How Devsisters built a world-class game with over 150 million downloads on CockroachDB

Why did we choose CockroachDB?

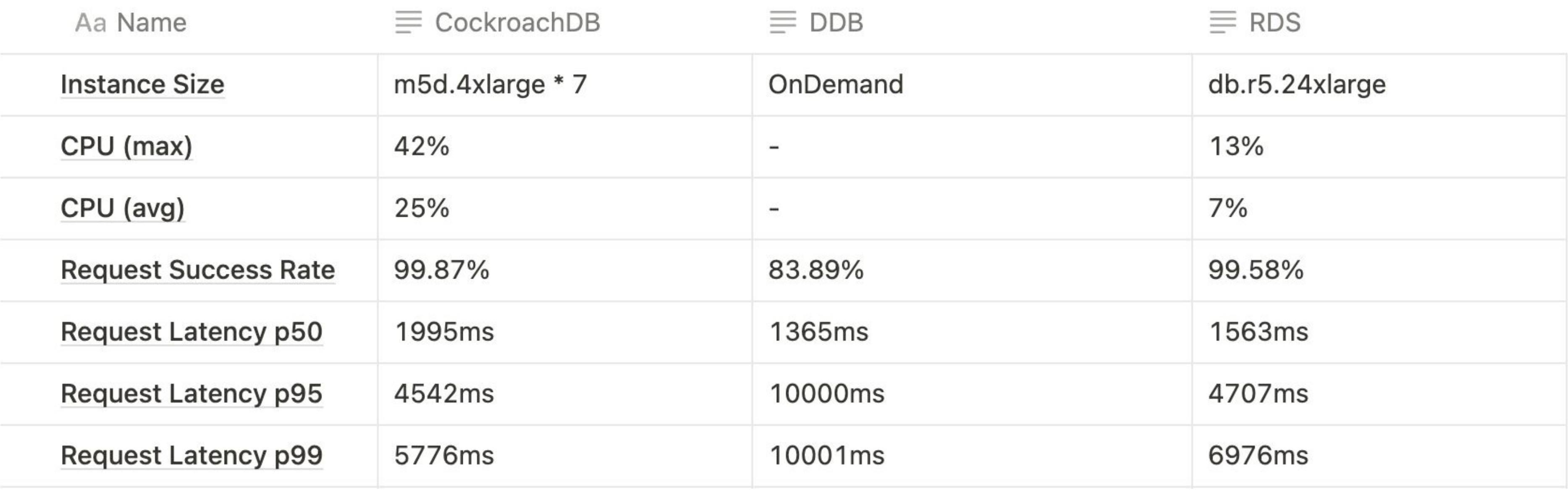
Before:

- Document–based database to support frequently changing schemas
- Distributed database for horizontal scalability

Evolving Needs:

- Support transactions
- Support extremely heavy traffic
- Ability to horizontally scale as needed
- Choose consistency over availability

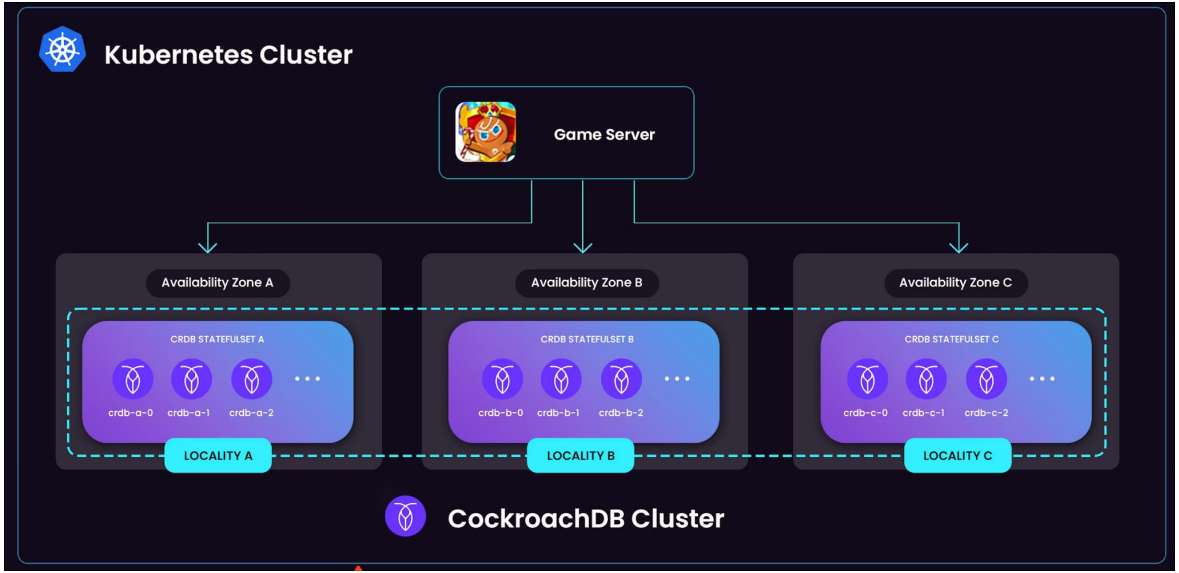
CRDB vs DDB vs RDS



Name	CockroachDB	DDB	RDS
Instance Size	m5d.4xlarge * 7	OnDemand	db.r5.24xlarge
CPU (max)	42%	-	13%
CPU (avg)	25%	-	7%
Request Success Rate	99.87%	83.89%	99.58%
Request Latency p50	1995ms	1365ms	1563ms
Request Latency p95	4542ms	10000ms	4707ms
Request Latency p99	5776ms	10001ms	6976ms

Disclaimer: This data dates back in 2020

CockroachDB on EKS



Monitoring CockroachDB

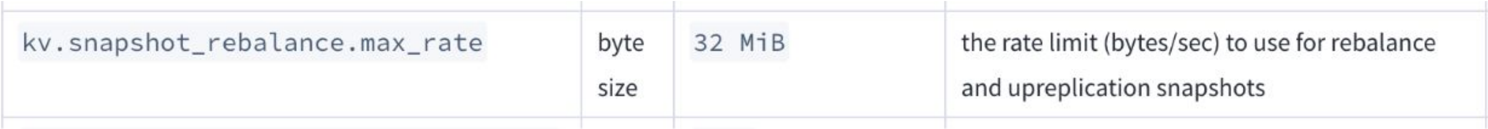
- Dashboards and monitors using Datadog
  - Datadog's AWS + CockroachDB Integrations
  - Can also be monitored with Prometheus

- Key metrics we keep eyes on
  - **USE metrics:** Basic dedicated instance usages
  - **EBS:** IOPS/Throughput, Queue Length, Write Stalls
  - **CockroachDB:** Unavailable ranges, Capacity

Operation Tips

Rolling CockroachDB nodes without downtime

- Rolling nodes is fine, but don’t go too fast
    - We had an outage when too many nodes were decommissioned at once
    - Our team only decommissions 3 nodes in same AZ at once
- Adjust rebalancing rate limits for faster rebalancing
    - Default of 32 MiB/s was too slow, we found 128 MiB/s to be optimal
    - Network, Disk I/O for rebalancing might interfere with workloads if too high

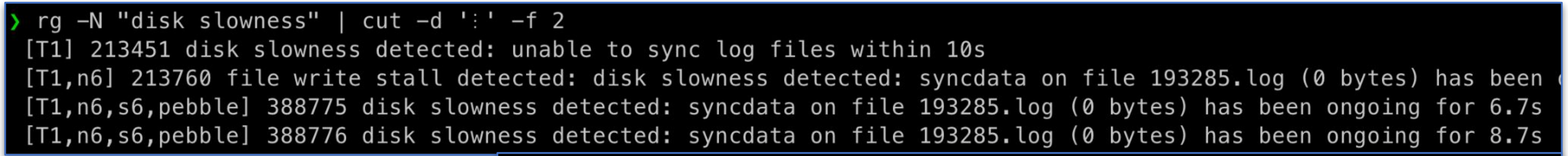


`kv.snapshot_rebalance.max_rate`	byte size	`32 MiB`	the rate limit (bytes/sec) to use for rebalance and upreplication snapshots

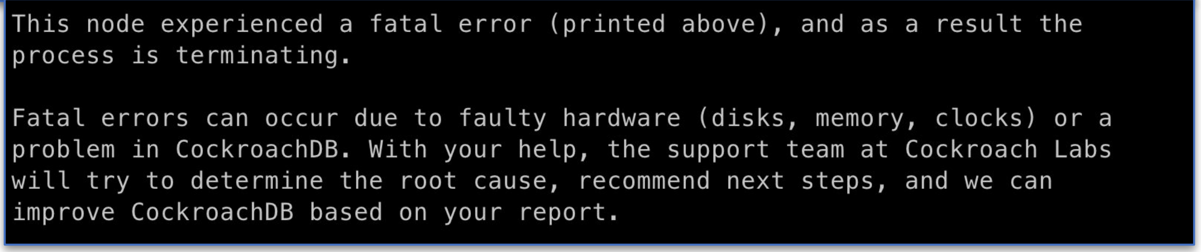
Operation Tips

EBS Hiccups

- I/O may stall intermittently for EBS volumes
  - Underlying machine for an EBS volume may failover – I/O will pause during failovers
  - CRDB node connected to the volume will crash and restart if this happens
- Services were unaffected for us, but this may be critical depending on workloads



```
❯ rg -N "disk slowness" | cut -d ':' -f 2
 [T1] 213451 disk slowness detected: unable to sync log files within 10s
 [T1,n6] 213760 file write stall detected: disk slowness detected: syncdata on file 193285.log (0 bytes) has been
 [T1,n6,s6,pebble] 388775 disk slowness detected: syncdata on file 193285.log (0 bytes) has been ongoing for 6.7s
 [T1,n6,s6,pebble] 388776 disk slowness detected: syncdata on file 193285.log (0 bytes) has been ongoing for 8.7s
```

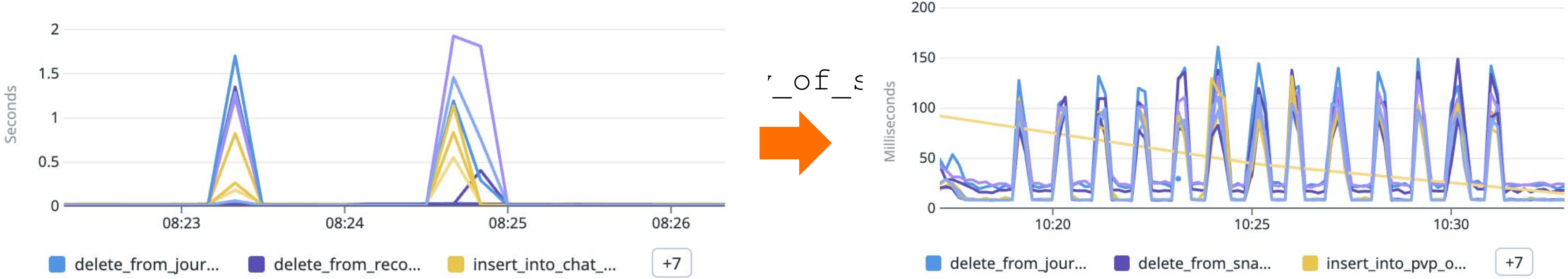


```
This node experienced a fatal error (printed above), and as a result the
process is terminating.

Fatal errors can occur due to faulty hardware (disks, memory, clocks) or a
problem in CockroachDB. With your help, the support team at Cockroach Labs
will try to determine the root cause, recommend next steps, and we can
improve CockroachDB based on your report.
```

Operation Tips

Admission control for analytics workloads

- ETL tasks need to be run daily on the database
  - Huge surge in SELECT queries was needed, but cluster was rightsized without this considered
- Set session level QoS for OLAP queries: CockroachDB will run other workloads before ETL



p95 Query Latency: **2s → 150ms**

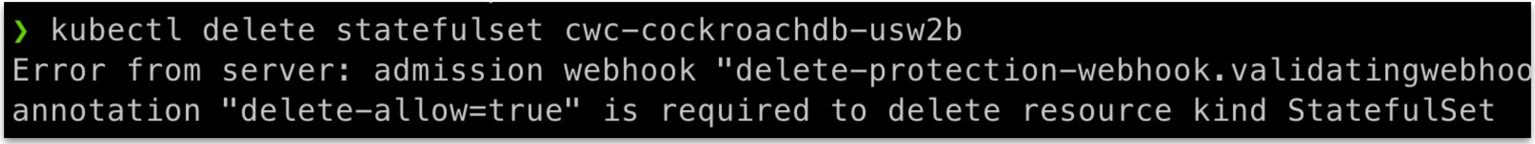
https://www.cockroachlabs.com/blog/admission-control-in-cockroachdb/

Highly Recommend Introducing A Fail-Preventing Webhook

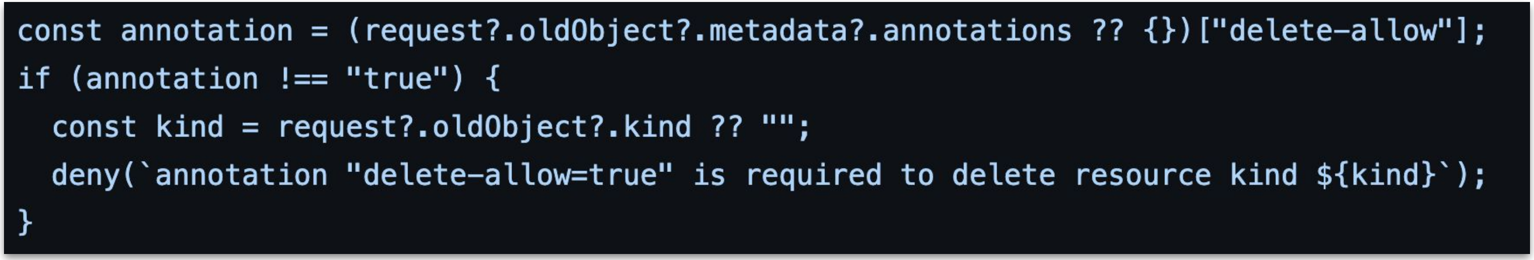
It’s easy to accidentally delete resources in Kubernetes..

- Use a Kubernetes admission webhook to block accidental operations
    - Webhook will deny StatefulSet, PV delete requests unless a specific annotation is set
- We use JavaScript to write webhooks – it’s open source!

  https://github.com/devsisters/checkpoint

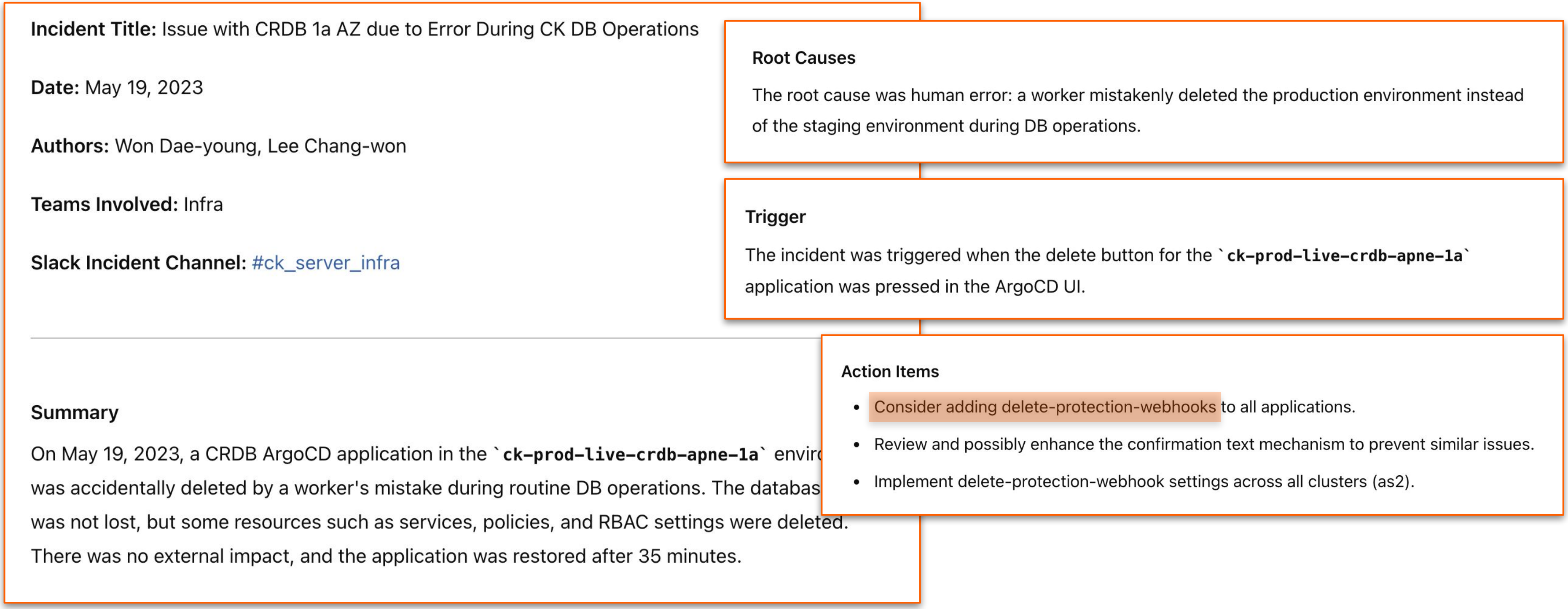


```
❯ kubectl delete statefulset cwc-cockroachdb-usw2b
Error from server: admission webhook "delete-protection-webhook.validatingwebhoo
annotation "delete-allow=true" is required to delete resource kind StatefulSet
```



```
const annotation = (request?.oldObject?.metadata?.annotations ?? {})["delete-allow"];
if (annotation !== "true") {
  const kind = request?.oldObject?.kind ?? "";
  deny(`annotation "delete-allow=true" is required to delete resource kind ${kind}`);
}
```

Highly Recommend Introducing A Fail-Preventing Webhook



Incident Title: Issue with CRDB 1a AZ due to Error During CK DB Operations

Date: May 19, 2023

Authors: Won Dae-young, Lee Chang-won

Teams Involved: Infra

Slack Incident Channel: #ck_server_infra

Summary

On May 19, 2023, a CRDB ArgoCD application in the `ck-prod-live-crdb-apne-1a` envir was accidentally deleted by a worker's mistake during routine DB operations. The databas was not lost, but some resources such as services, policies, and RBAC settings were deleted. There was no external impact, and the application was restored after 35 minutes.

Root Causes

The root cause was human error: a worker mistakenly deleted the production environment instead of the staging environment during DB operations.

Trigger

The incident was triggered when the delete button for the `ck-prod-live-crdb-apne-1a` application was pressed in the ArgoCD UI.

Action Items

- Consider adding delete-protection-webhooks to all applications.
- Review and possibly enhance the confirmation text mechanism to prevent similar issues.
- Implement delete-protection-webhook settings across all clusters (as2).

Characteristics of Mobile Game Development

- Each engineer often works on isolated features
    - Server developers, client developers, QA engineers, …
- Database state highly affects developers’ workflow
- Even multiple production environments are required
    - App Store review, press events, on-site game shows, …

Characteristics of Mobile Game Development

- Each engineer often works on isolated features
  - Server developers, client developers, QA engineers, …
- Database state highly affects developers' workflow
- Even multiple production environments are required
  - App Store review, press events, on-site game shows, …

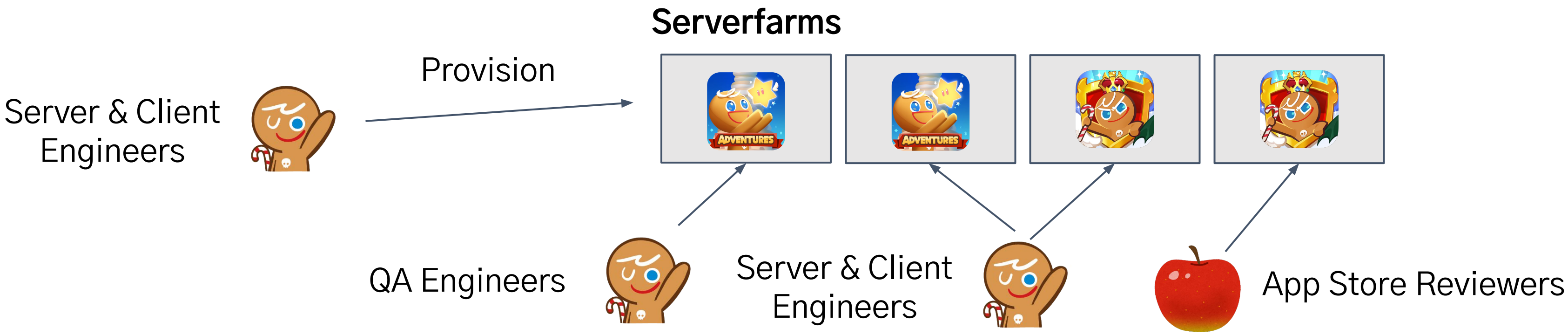
Developers need to easily spin up isolated server deployments

Serverfarm

Definition: Isolated server applications that are reproducible and fully operational by itself

Technical Requirements

- **Reproducible:** Should be declaratively defined
- **Isolation:** No data or network traffic should flow between serverfarms
- **Self-service:** Easily configured & provisioned on demand by engineers



Serverfarms

Provision

Server & Client Engineers

QA Engineers

Server & Client Engineers

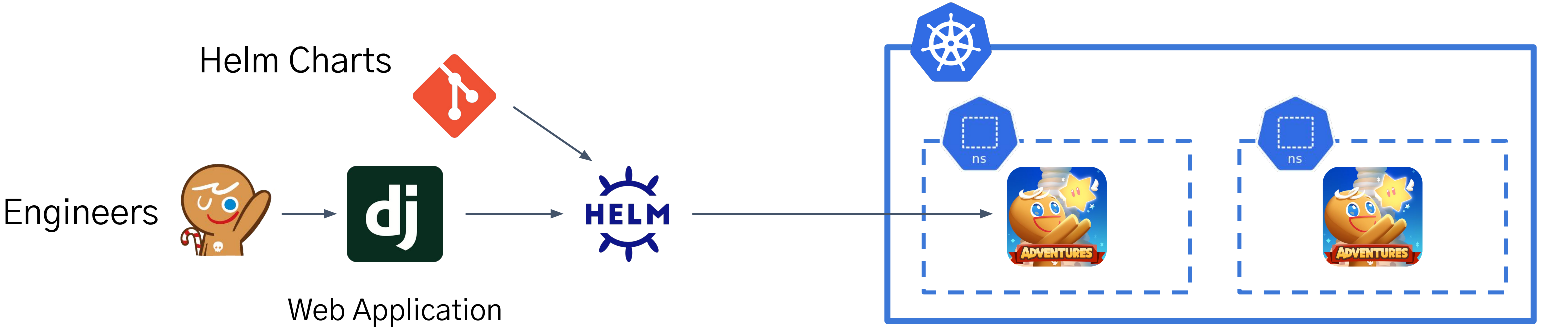
App Store Reviewers

Serverfarm

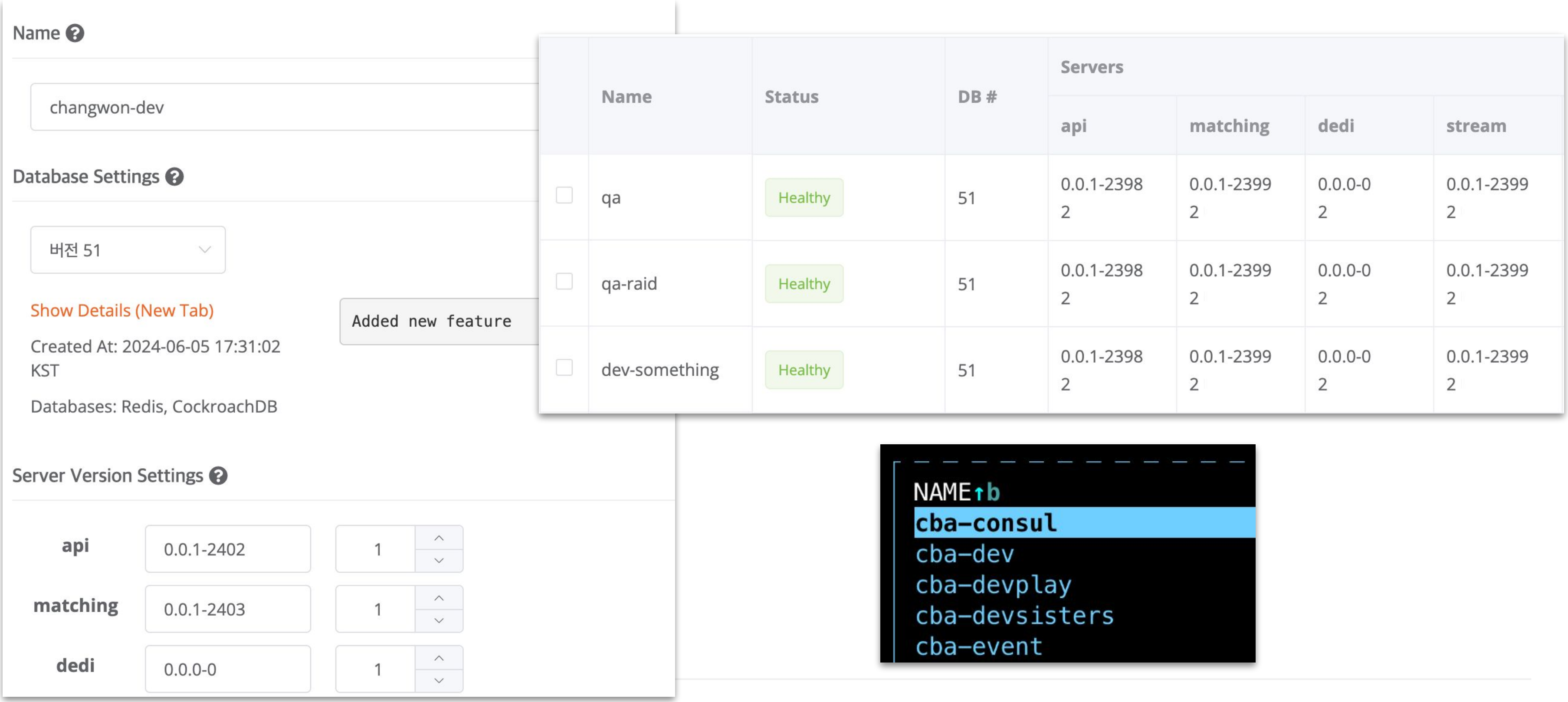
Kubernetes solves these problems really well!

Technical Requirements

- **Reproducible:** Helm charts are reproducible and declarative
- **Isolation:** Kubernetes namespaces isolate workloads
- **Self-service:** Wrote a light web application that runs `helm install`



Deploying ‘serverfarms’



Name

changwon-dev

Database Settings

버전 51

Show Details (New Tab)

Added new feature

Created At: 2024-06-05 17:31:02 KST

Databases: Redis, CockroachDB

Server Version Settings

api	0.0.1-2402	1
matching	0.0.1-2403	1
dedi	0.0.0-0	1

	Name	Status	DB #	Servers			
				api	matching	dedi	stream
□	qa	Healthy	51	0.0.1-2398 2	0.0.1-2399 2	0.0.0-0 2	0.0.1-2399 2
□	qa-raid	Healthy	51	0.0.1-2398 2	0.0.1-2399 2	0.0.0-0 2	0.0.1-2399 2
□	dev-something	Healthy	51	0.0.1-2398 2	0.0.1-2399 2	0.0.0-0 2	0.0.1-2399 2

```
NAME↑b
cba-consul
cba-dev
cba-devplay
cba-devsisters
cba-event
```

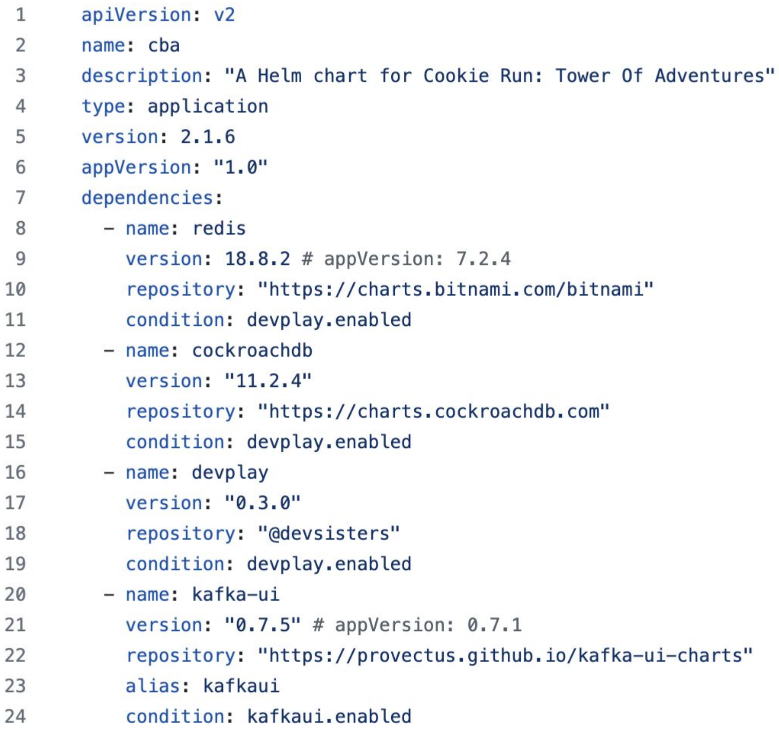
Serverfarms and CockroachDB

CockroachDB is also deployed as a Helm chart

Benefits of deploying CockroachDB as a chart

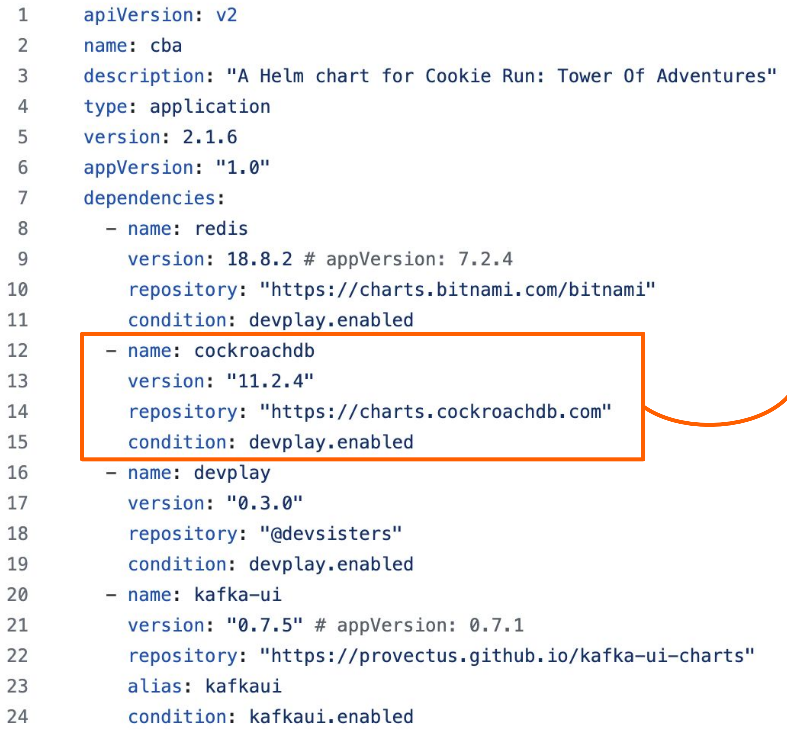
- **Flexible:** Works seamlessly with all our use cases
  - Development, QA: Installed as a game server chart dependency
  - Production: Managed safely with ArgoCD
- **Automated:** OIDC configuration, pre-splitting tables, etc. are scripted
- **Reusable:** CockroachDB configurations are written in code to be reusable
  - Knowledge is passed down throughout multiple projects

Deploying CockroachDB



```yaml
apiVersion: v2
name: cba
description: "A Helm chart for Cookie Run: Tower Of Adventures"
type: application
version: 2.1.6
appVersion: "1.0"
dependencies:
  - name: redis
    version: 18.8.2 # appVersion: 7.2.4
    repository: "https://charts.bitnami.com/bitnami"
    condition: devplay.enabled
  - name: cockroachdb
    version: "11.2.4"
    repository: "https://charts.cockroachdb.com"
    condition: devplay.enabled
  - name: devplay
    version: "0.3.0"
    repository: "@devsisters"
    condition: devplay.enabled
  - name: kafka-ui
    version: "0.7.5" # appVersion: 0.7.1
    repository: "https://provectus.github.io/kafka-ui-charts"
    alias: kafkaui
    condition: kafkaui.enabled
```

Deploying CockroachDB



```yaml
apiVersion: v2
name: cba
description: "A Helm chart for Cookie Run: Tower Of Adventures"
type: application
version: 2.1.6
appVersion: "1.0"
dependencies:
  - name: redis
    version: 18.8.2 # appVersion: 7.2.4
    repository: "https://charts.bitnami.com/bitnami"
    condition: devplay.enabled
  - name: cockroachdb
    version: "11.2.4"
    repository: "https://charts.cockroachdb.com"
    condition: devplay.enabled
  - name: devplay
    version: "0.3.0"
    repository: "@devsisters"
    condition: devplay.enabled
  - name: kafka-ui
    version: "0.7.5" # appVersion: 0.7.1
    repository: "https://provectus.github.io/kafka-ui-charts"
    alias: kafkaui
    condition: kafkaui.enabled
```

CRDB cluster ready? ✅

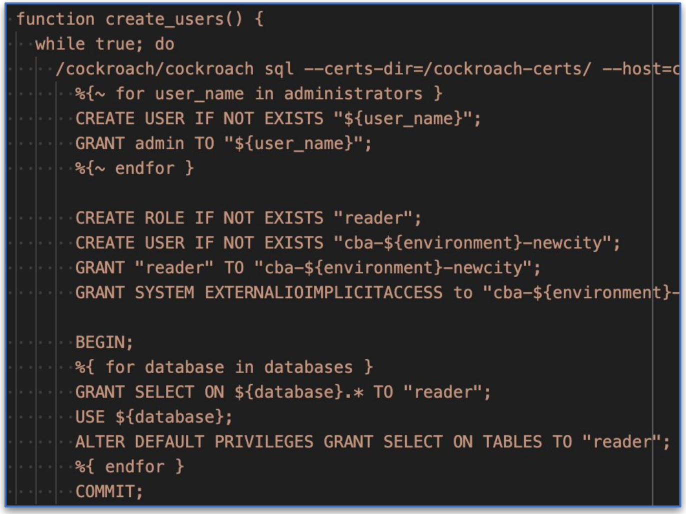
Application ready? 🤔

No!

Need to initialize the database with schema and cluster settings

Automation

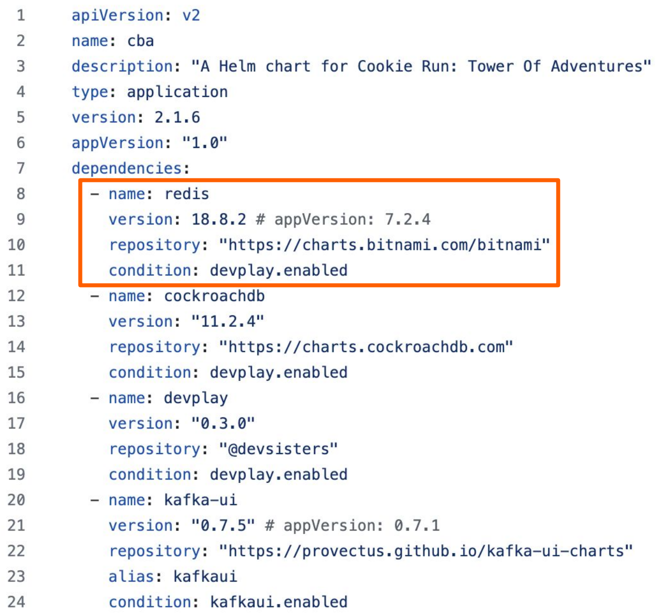
- Use Kubernetes Jobs to run initialization script
  - Initialize cluster
  - Integrate OIDC with our identity provider
  - Apply CRDB cluster settings
  - Create backup jobs (full and incremental)
  - Create users, roles and grant privileges
- Shell script embedded in our Helm chart



```
function create_users() {
  while true; do
    /cockroach/cockroach sql --certs-dir=/cockroach-certs/ --host=c
      %{~ for user_name in administrators }
      CREATE USER IF NOT EXISTS "${user_name}";
      GRANT admin TO "${user_name}";
      %{~ endfor }

      CREATE ROLE IF NOT EXISTS "reader";
      CREATE USER IF NOT EXISTS "cba-${environment}-newcity";
      GRANT "reader" TO "cba-${environment}-newcity";
      GRANT SYSTEM EXTERNALIOIMPLICITACCESS to "cba-${environment}-

      BEGIN;
      %{ for database in databases }
      GRANT SELECT ON ${database}.* TO "reader";
      USE ${database};
      ALTER DEFAULT PRIVILEGES GRANT SELECT ON TABLES TO "reader";
      %{ endfor }
      COMMIT;
```

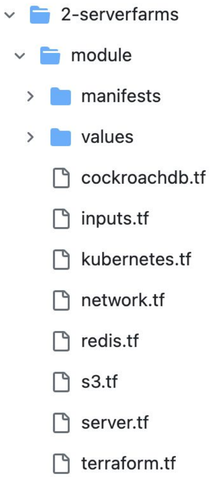
Terraform

- Some components are better to use managed services
- STG/PROD environments are more rigid than DEV

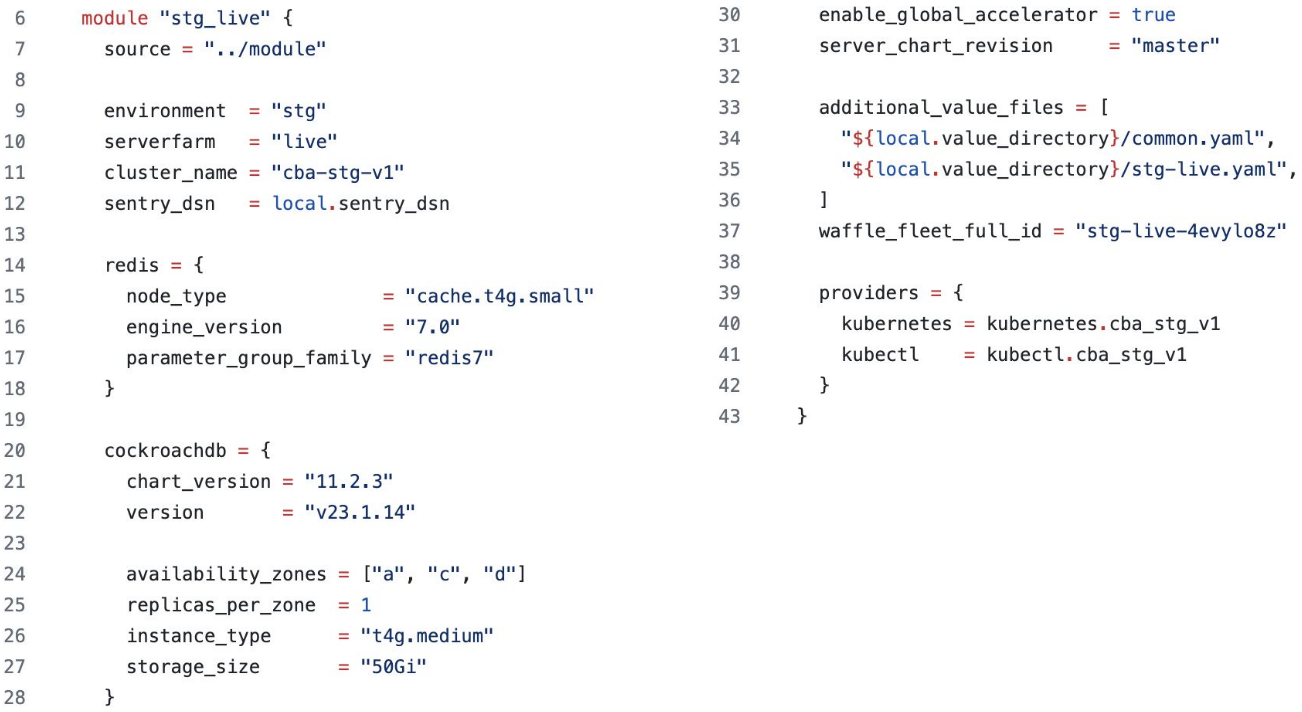


```yaml
apiVersion: v2
name: cba
description: "A Helm chart for Cookie Run: Tower Of Adventures"
type: application
version: 2.1.6
appVersion: "1.0"
dependencies:
  - name: redis
    version: 18.8.2 # appVersion: 7.2.4
    repository: "https://charts.bitnami.com/bitnami"
    condition: devplay.enabled
  - name: cockroachdb
    version: "11.2.4"
    repository: "https://charts.cockroachdb.com"
    condition: devplay.enabled
  - name: devplay
    version: "0.3.0"
    repository: "@devsisters"
    condition: devplay.enabled
  - name: kafka-ui
    version: "0.7.5" # appVersion: 0.7.1
    repository: "https://provectus.github.io/kafka-ui-charts"
    alias: kafkaui
    condition: kafkaui.enabled
```

Terraform



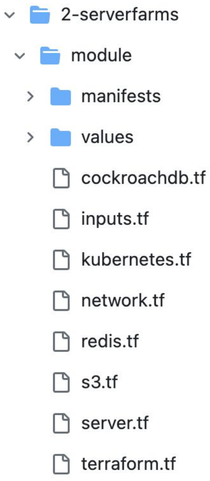
- 2-serverfarms
  - module
    - manifests
    - values
    - cockroachdb.tf
    - inputs.tf
    - kubernetes.tf
    - network.tf
    - redis.tf
    - s3.tf
    - server.tf
    - terraform.tf



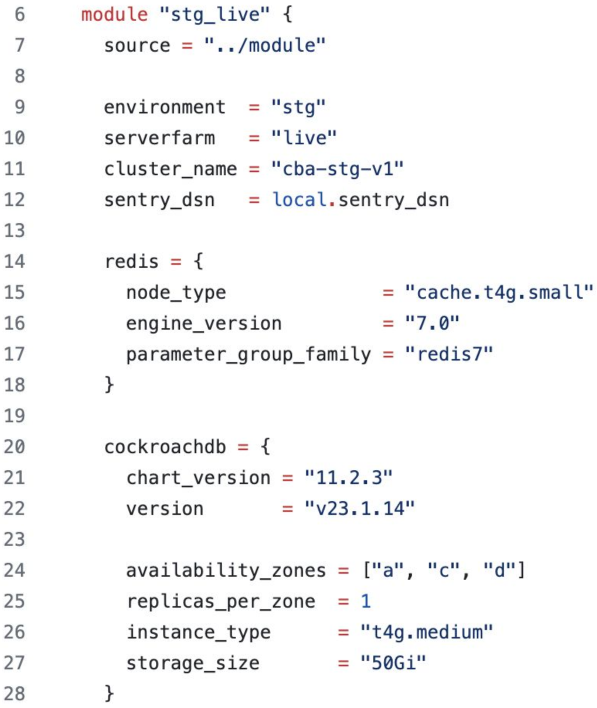
```
module "stg_live" {
  source = "../module"

  environment  = "stg"
  serverfarm   = "live"
  cluster_name = "cba-stg-v1"
  sentry_dsn   = local.sentry_dsn

  redis = {
    node_type              = "cache.t4g.small"
    engine_version         = "7.0"
    parameter_group_family = "redis7"
  }

  cockroachdb = {
    chart_version = "11.2.3"
    version       = "v23.1.14"

    availability_zones = ["a", "c", "d"]
    replicas_per_zone  = 1
    instance_type      = "t4g.medium"
    storage_size       = "50Gi"
  }
```

```
  enable_global_accelerator = true
  server_chart_revision     = "master"

  additional_value_files = [
    "${local.value_directory}/common.yaml",
    "${local.value_directory}/stg-live.yaml",
  ]
  waffle_fleet_full_id = "stg-live-4evylo8z"

  providers = {
    kubernetes = kubernetes.cba_stg_v1
    kubectl    = kubectl.cba_stg_v1
  }
}
```

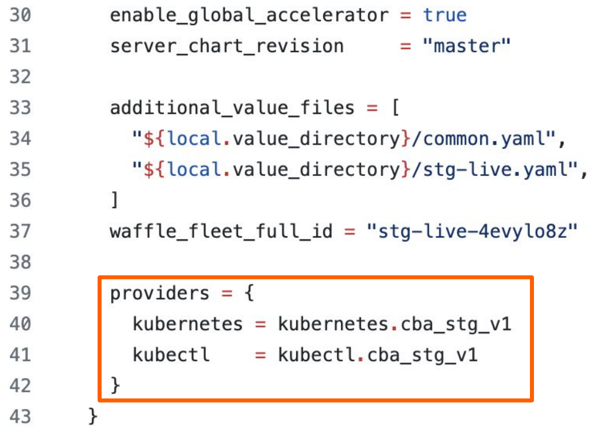
Terraform



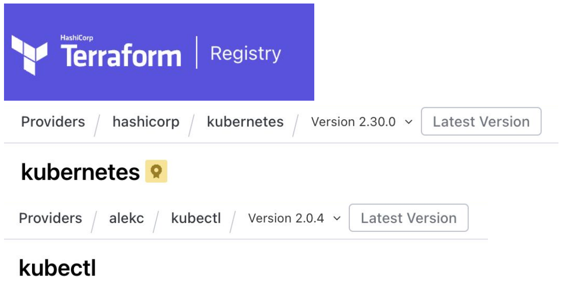
- 2-serverfarms
  - module
    - manifests
    - values
    - cockroachdb.tf
    - inputs.tf
    - kubernetes.tf
    - network.tf
    - redis.tf
    - s3.tf
    - server.tf
    - terraform.tf



```
module "stg_live" {
  source = "../module"

  environment  = "stg"
  serverfarm   = "live"
  cluster_name = "cba-stg-v1"
  sentry_dsn   = local.sentry_dsn

  redis = {
    node_type              = "cache.t4g.small"
    engine_version         = "7.0"
    parameter_group_family = "redis7"
  }

  cockroachdb = {
    chart_version = "11.2.3"
    version       = "v23.1.14"

    availability_zones = ["a", "c", "d"]
    replicas_per_zone  = 1
    instance_type      = "t4g.medium"
    storage_size       = "50Gi"
  }
```



```
  enable_global_accelerator = true
  server_chart_revision     = "master"

  additional_value_files = [
    "${local.value_directory}/common.yaml",
    "${local.value_directory}/stg-live.yaml",
  ]
  waffle_fleet_full_id = "stg-live-4evylo8z"

  providers = {
    kubernetes = kubernetes.cba_stg_v1
    kubectl    = kubectl.cba_stg_v1
  }
}
```



HashiCorp Terraform | Registry

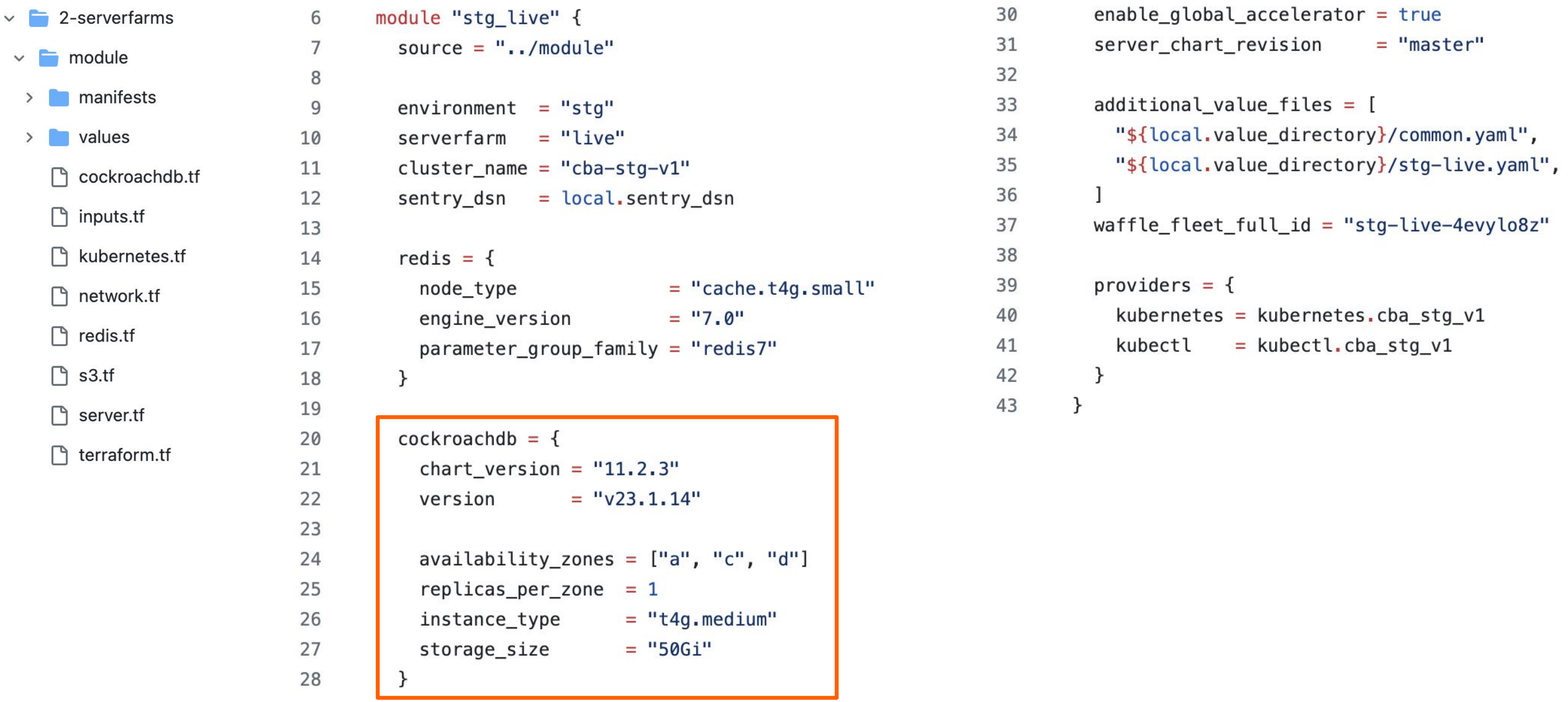
Providers / hashicorp / kubernetes Version 2.30.0 Latest Version

kubernetes

Providers / alekc / kubectl Version 2.0.4 Latest Version

kubectl

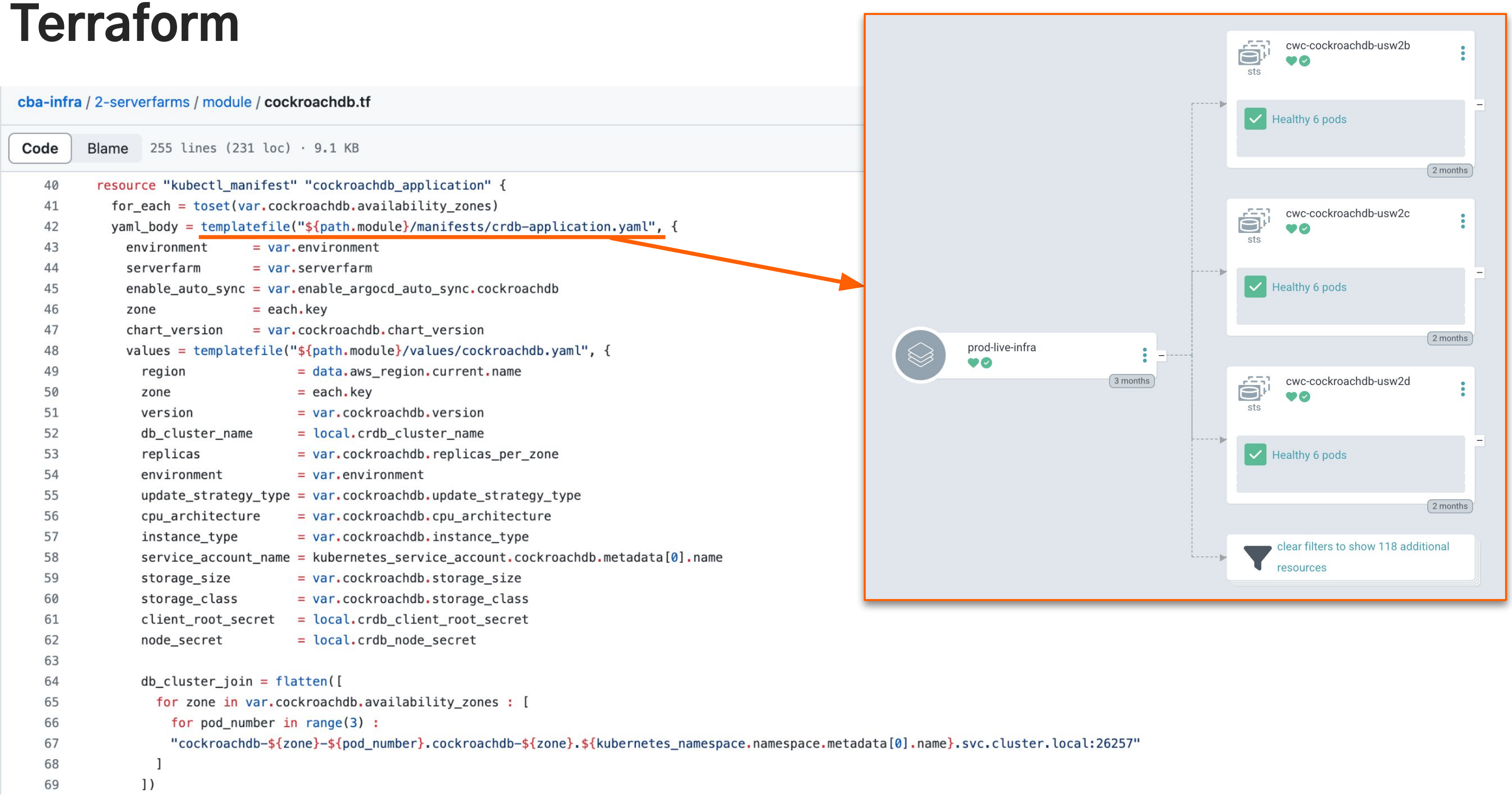
Terraform



```
2-serverfarms
  module
    manifests
    values
    cockroachdb.tf
    inputs.tf
    kubernetes.tf
    network.tf
    redis.tf
    s3.tf
    server.tf
    terraform.tf
```

```
module "stg_live" {
  source = "../module"

  environment  = "stg"
  serverfarm   = "live"
  cluster_name = "cba-stg-v1"
  sentry_dsn   = local.sentry_dsn

  redis = {
    node_type              = "cache.t4g.small"
    engine_version         = "7.0"
    parameter_group_family = "redis7"
  }

  cockroachdb = {
    chart_version = "11.2.3"
    version       = "v23.1.14"

    availability_zones = ["a", "c", "d"]
    replicas_per_zone  = 1
    instance_type      = "t4g.medium"
    storage_size       = "50Gi"
  }
  enable_global_accelerator = true
  server_chart_revision     = "master"

  additional_value_files = [
    "${local.value_directory}/common.yaml",
    "${local.value_directory}/stg-live.yaml",
  ]
  waffle_fleet_full_id = "stg-live-4evylo8z"

  providers = {
    kubernetes = kubernetes.cba_stg_v1
    kubectl    = kubectl.cba_stg_v1
  }
}
```

Terraform



cba-infra / 2-serverfarms / module / cockroachdb.tf

Code Blame 255 lines (231 loc) · 9.1 KB

```
resource "kubectl_manifest" "cockroachdb_application" {
  for_each = toset(var.cockroachdb.availability_zones)
  yaml_body = templatefile("${path.module}/manifests/crdb-application.yaml", {
    environment      = var.environment
    serverfarm       = var.serverfarm
    enable_auto_sync = var.enable_argocd_auto_sync.cockroachdb
    zone             = each.key
    chart_version    = var.cockroachdb.chart_version
    values = templatefile("${path.module}/values/cockroachdb.yaml", {
      region               = data.aws_region.current.name
      zone                 = each.key
      version              = var.cockroachdb.version
      db_cluster_name      = local.crdb_cluster_name
      replicas             = var.cockroachdb.replicas_per_zone
      environment          = var.environment
      update_strategy_type = var.cockroachdb.update_strategy_type
      cpu_architecture     = var.cockroachdb.cpu_architecture
      instance_type        = var.cockroachdb.instance_type
      service_account_name = kubernetes_service_account.cockroachdb.metadata[0].name
      storage_size         = var.cockroachdb.storage_size
      storage_class        = var.cockroachdb.storage_class
      client_root_secret   = local.crdb_client_root_secret
      node_secret          = local.crdb_node_secret

      db_cluster_join = flatten([
        for zone in var.cockroachdb.availability_zones : [
          for pod_number in range(3) :
          "cockroachdb-${zone}-${pod_number}.cockroachdb-${zone}.${kubernetes_namespace.namespace.metadata[0].name}.svc.cluster.local:26257"
        ]
      ])
```

clear filters to show 118 additional resources

Don’t reinvent the wheel!

- How did we internalize this principle?
  - Learned while working with Tencent :P

- Engineer’s dilemma: We have a tendency to solve problems in fancy and complex ways

- Complexity comes with cost of learning
  - Simple and straightforward solutions enhance collaboration and efficiency

Key Points Recap

- **Kubernetes** as a baseline of infrastructure deployment
  - Agnostic of cloud providers and regions
  - **Make an error-preventing Kubernetes webhook!**
- Abstraction of application deployment using common tools
  - **Helm charts** to define ad-hoc, lightweight environments (DEV)
    - Kubernetes Jobs executing initialization shell scripts (cluster init, settings, users & privileges)
  - **Terraform** and **ArgoCD** for more rigid, serious environments (STG / PROD)
- **CockroachDB** is designed to fit in cloud-native operations!
- **Make everything "*boring*" to easily communicate with other teams!**

Pre-register Now!

Cookie-crisp Top-down action!
CookieRun
TOWER OF ADVENTURES
GRAND LAUNCH
2024.06.25
Pre-register in your app marketplace to get Chamomile Cookie!
Pre-order on the App Store
PRE-REGISTER ON Google Play
CookieRun: Tower of Adventures will be officially available through Google Play Games on PC

Thank you!

ChangWon Lee

Lead DevOps Engineer @ Devsisters

changwon.lee@devsisters.com