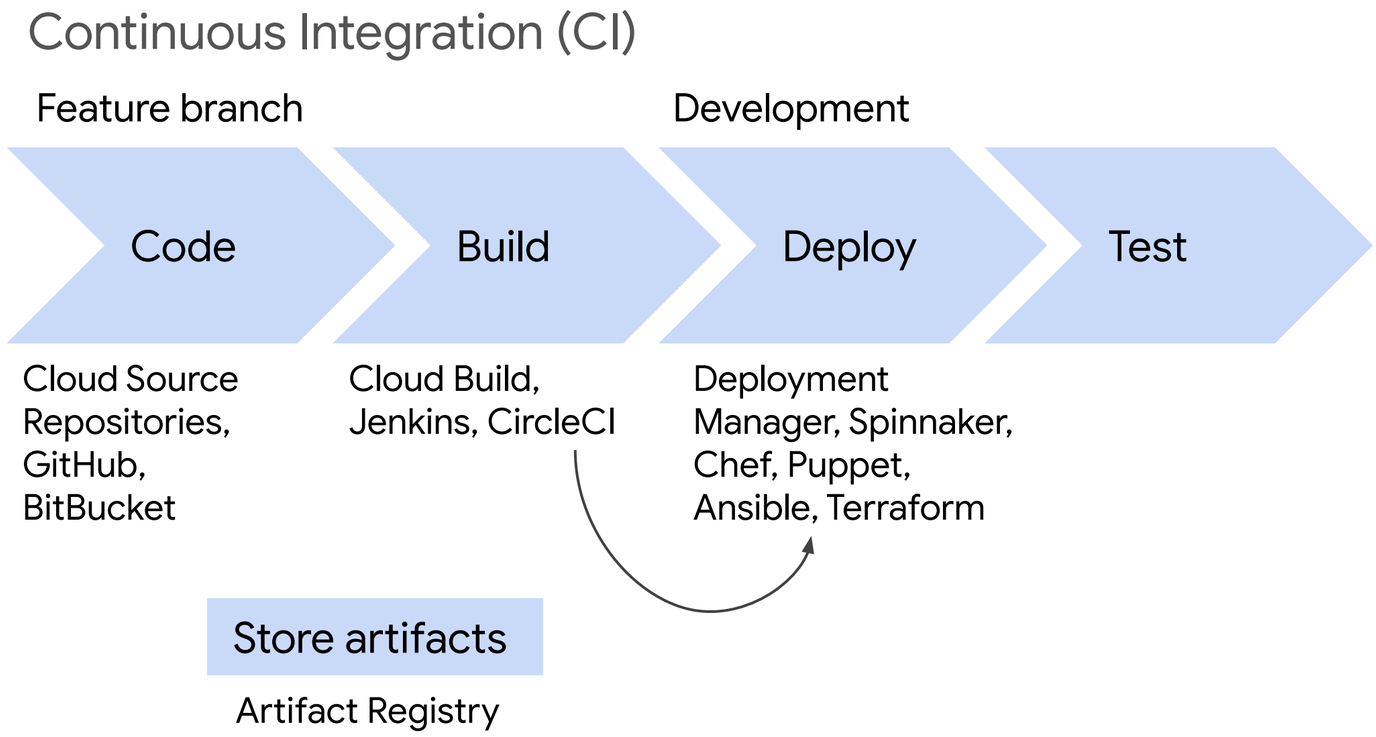
Google Cloud

DevOps Automation

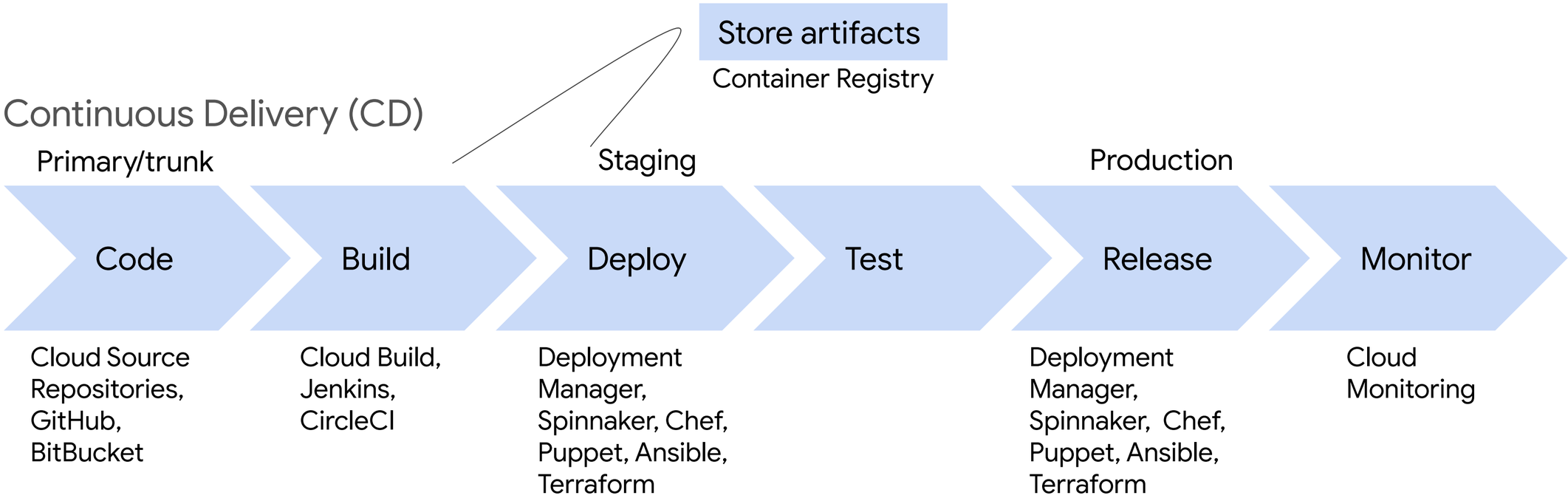
The following course materials are copyright protected materials.

They may not be reproduced or distributed and may only be used by students attending this Google Cloud Partner Learning Services program.

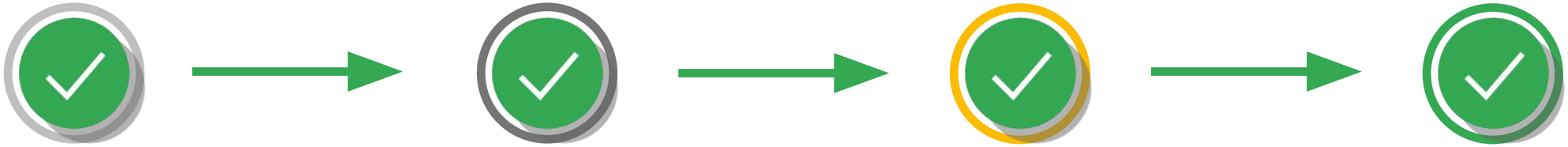
Implement continuous integration and delivery for reliable releases



Implement continuous integration and delivery for reliable releases



Continuous integration pipelines automate building applications



Developers check-in code

Use a Git repo for each microservice and branches for versions.

Run unit tests

If the tests don't pass, stop.

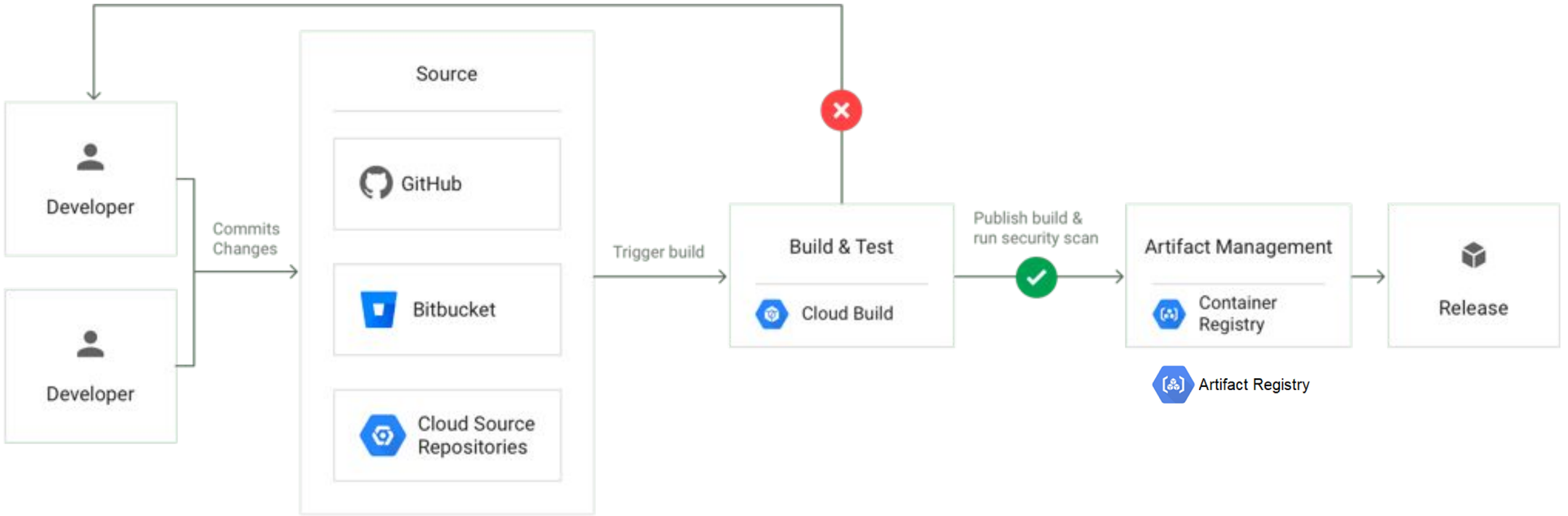
Build deployment package

Create a Docker image.

Deploy

Save your new Docker image in a container registry.

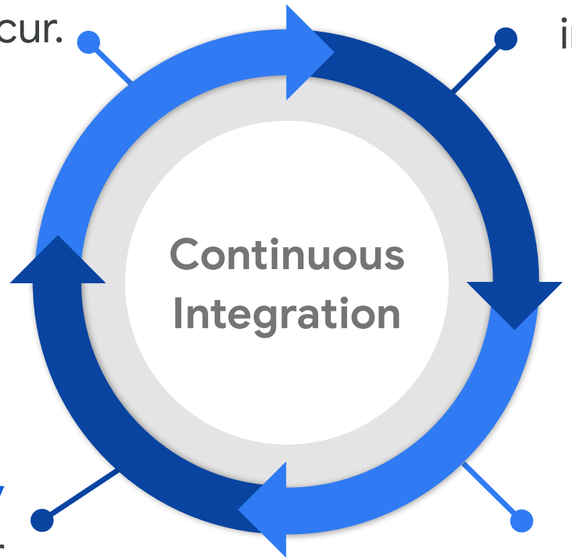
CI/CD - a closer look



Google provides the components required for a continuous integration pipeline

Cloud Source Repositories
Developers push to a central repository when they want a build to occur.

Cloud Build
Build system executes the steps required to make a deployment package or Docker image.

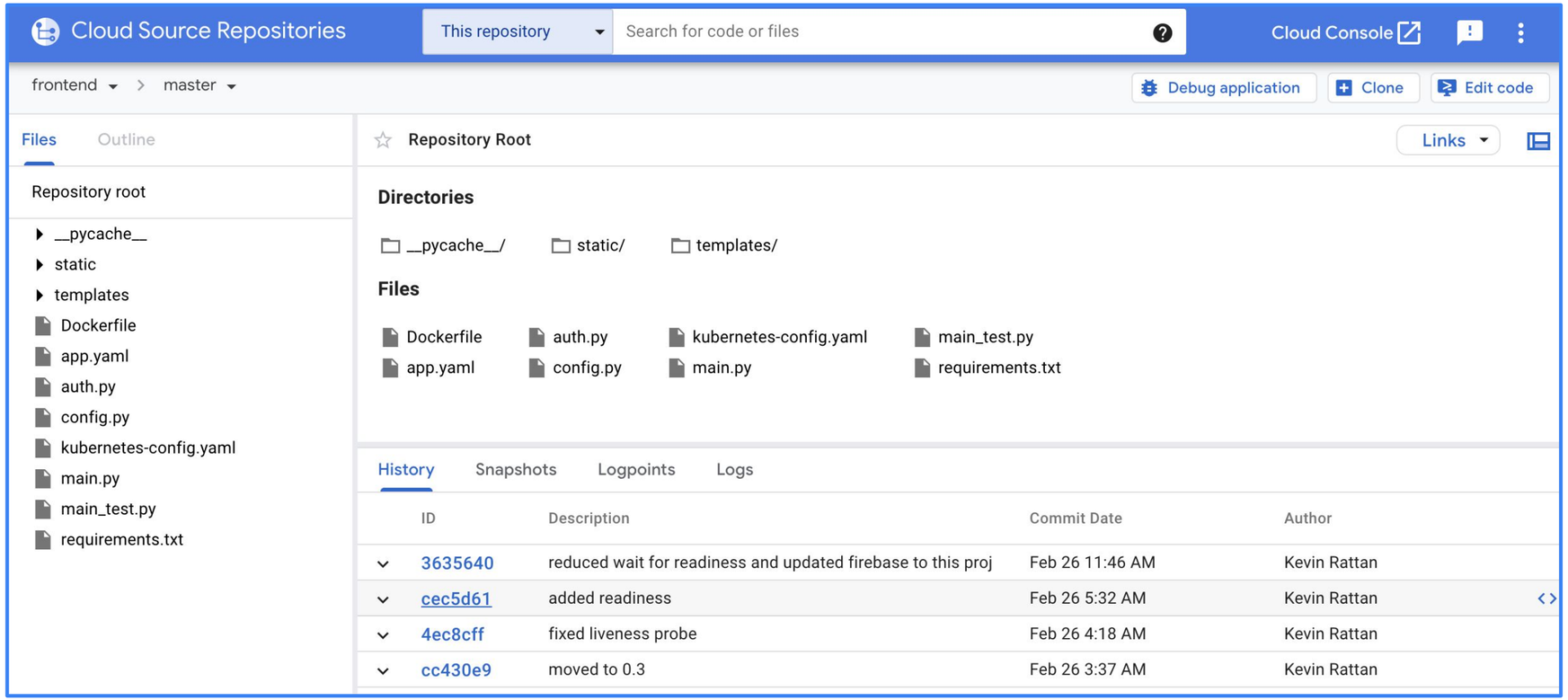


Artifact Registry
Store your Docker images or deployment packages in a central location for deployment.

Build triggers
Watches for changes in the Git repo and starts the build.

Cloud Source Repositories provides managed Git repositories

Control access to your repos using IAM within your Google Cloud projects.



Google Cloud

Cloud Source Repository - IAM Roles/Key detection

Access to repositories is controlled by granting appropriate roles to members (i.e. users, groups or service accounts e.g. Cloud Build Service Account)

- **roles/source.reader** Source Repository Reader
- **roles/source.writer** Source Repository Writer
- **roles/source.admin** Source Repository Administrator
  See https://cloud.google.com/source-repositories/docs/configure-access-control

A useful security feature in Cloud Source Repository is the **automatic detection of security keys in source code** (preventing inadvertent exposures in code artifacts)
https://cloud.google.com/source-repositories/docs/detecting-security-keys

Cloud Source Repository - Notifications

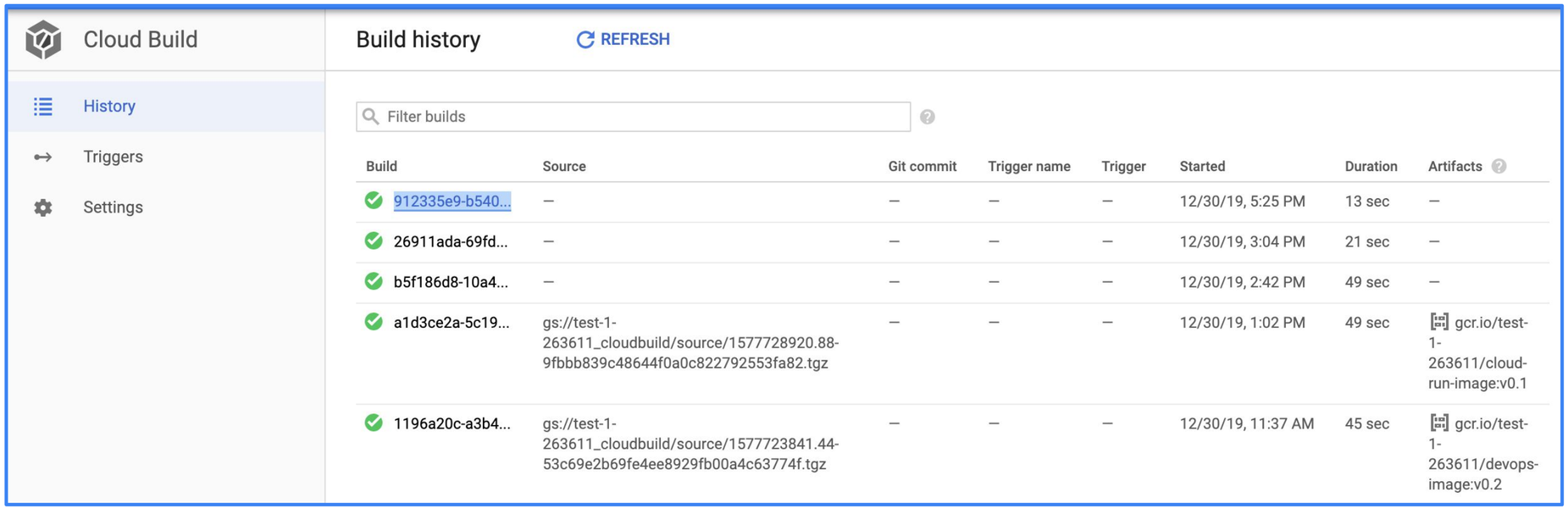
- You can configure Cloud Source Repository (CSR) to write events **to a Pub/Sub topic** when a repo event occurs e.g. create repo, push, delete repo. This enables programs to be started in response to the event e.g. notify an administrator of a new commit etc

https://cloud.google.com/source-repositories/docs/pubsub-notifications

- **Triggers** can also be used to start a cloud build every time there is a commit to a CSR repo. This is a fundamental part of automating the CI/CD pipeline

Cloud Build lets you build software quickly across all languages

- Google-hosted Docker build service
  - Alternative to using Docker build command
- Use the CLI to submit a build
  gcloud builds submit --tag gcr.io/your-project-id/image-name .



Cloud Build - features

Build - containers, apps (node.js, Java JARs, Go binaries) and VM images

Deploy - to GKE, GCE, Cloud Run, App Engine, Cloud Functions etc

Automate CI/CD - with triggers

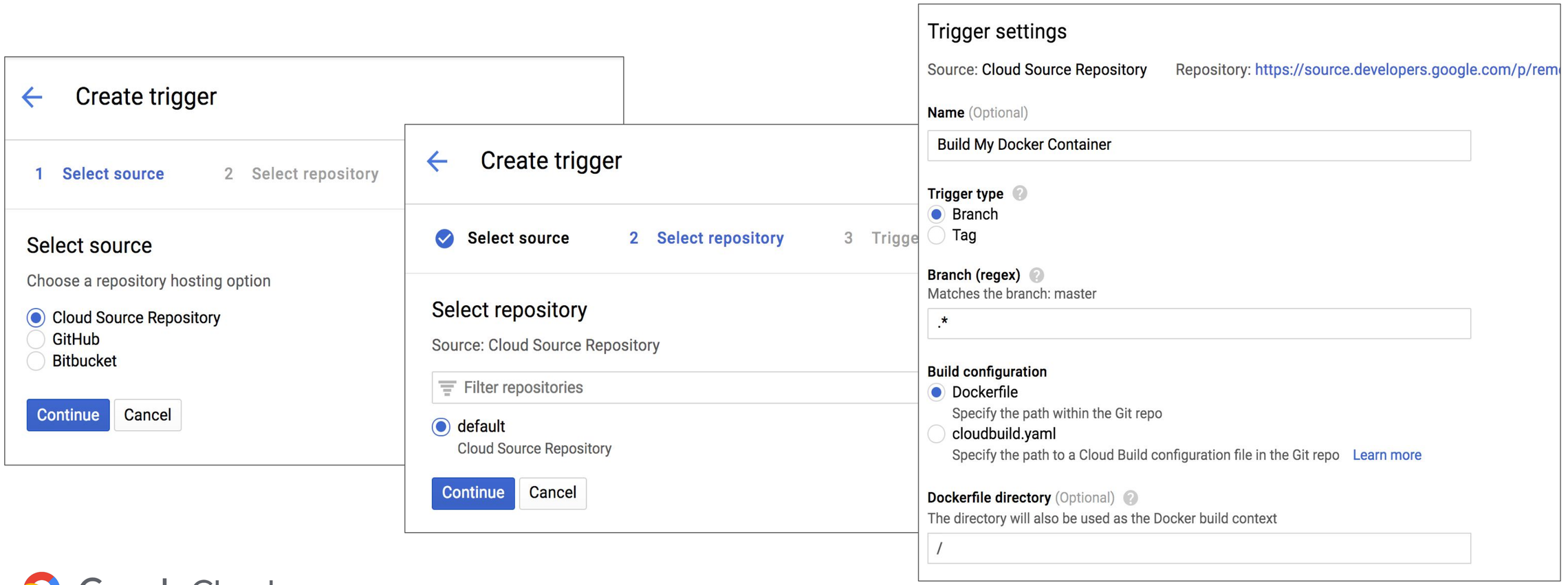
Store - artifacts in Artifact Registry or Cloud Storage (e.g. for binaries or tar balls)

Build configuration file specifies the steps in the build

Can speed up builds by e.g. requesting more powerful machines on build

Build triggers watch a repository and build a container whenever code is pushed

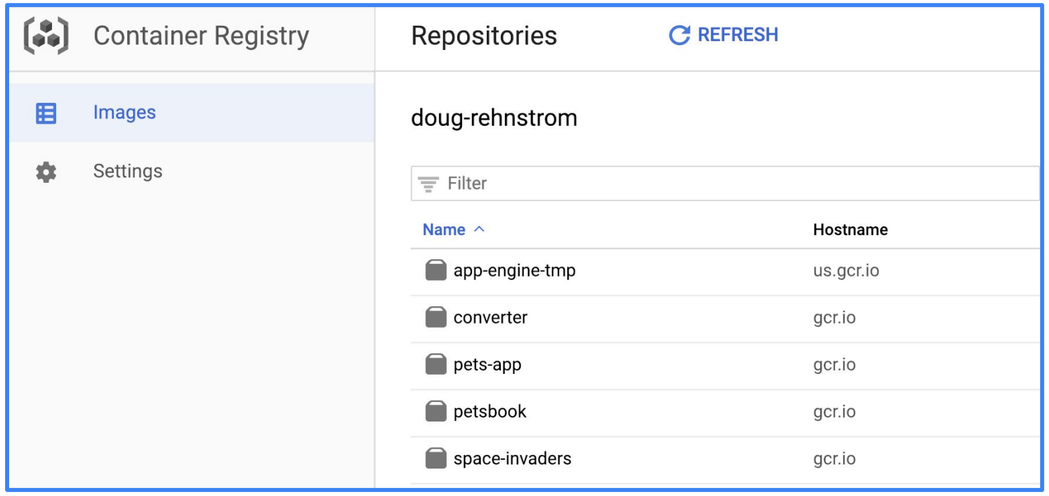
Supports Maven, custom builds, and Docker



Google Cloud

Container Registry is a Google Cloud–hosted Docker repository

- Images built using Cloud Build are automatically saved in Container Registry.
  - Tag images with the prefix **gcr.io/your-project-id**/image-name
- Can use Docker push and pull commands with Container Registry.
  - docker push gcr.io/your-project-id/image-name
  - docker pull gcr.io/your-project-id/image-name



Artifact Registry is the next generation of Container Registry

- Store universal build artifacts (not just Docker images)
- Deploy to all compute platforms e.g. GKE, GCE, Cloud Run, GAE, Cloud Functions
- Regional repositories
- Integration with other Google tools, e.g. Cloud Build, Source Repository etc
- Secure with IAM, Vulnerability Scanning, Binary Authorization, VPC Service Controls etc

https://cloud.google.com/artifact-registry/docs/overview

Introduction to Artifact Registry (YouTube)

Google Cloud Deploy

Deliver continuously to Google Kubernetes Engine.

- ✓ Create deployment pipelines for GKE within minutes
- ✓ Fully managed continuous delivery service for easy scaling
- ✓ Enterprise security and audit
- ✓ Built-in delivery metrics
- ✓ Snaps into your existing DevOps ecosystem

Google Cloud

Deploy tooling will vary

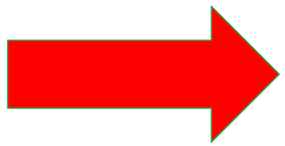
Deploy tooling will vary depending on the nature of the Compute platform e.g.

- Deploy on App Engine via **gcloud app deploy**
- Deploy on Cloud Run via **gcloud run deploy**
- Deploy in GKE by updating the deployment.yaml file and **kubectl apply -f**
- May also want other deployment capabilities, such as blue-green deployments, canarying etc

  Deployment steps can be automated with a variety of tools e.g Cloud Build, Spinnaker, Jenkins etc

Deploying to K8s - no automation

1. Change the source code
2. Build the new container image
3. Push the image to Artifact Registry
4. Update Deployment YAML and **kubectl apply** ..
5. Test if OK (with canarying? Blue/Green?)



Manual, error prone, toil - want to automate where possible/reasonable to do so!

Question

Question

What Google Cloud feature would be easiest to use to automate a build in response to code being checked into your source code repository?

A. Build triggers

B. Cloud Functions

C. App Engine

D. Cloud Scheduler

Question

Answer

What Google Cloud feature would be easiest to use to automate a build in response to code being checked into your source code repository?

A. Build triggers 

B. Cloud Functions

C. App Engine

D. Cloud Scheduler

More resources

Google Cloud DevOps solutions

https://cloud.google.com/devops/

CI/CD Automation

Security, Spinnaker, and IaC

Google Cloud

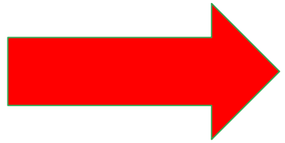
Securing CI/CD

- Managing secrets (e.g. passwords, access tokens etc)
- Securing build artifacts (e.g. container analysis)
- Controlling deployable images (binary authorization)

Managing secrets in the CI/CD process

Options include:

- Secret Manager in Google Cloud
- K8s Secrets
- Using K8s Secrets with extra KMS-based encryption
- Third party secret management tools e.g. Hashicorp Vault



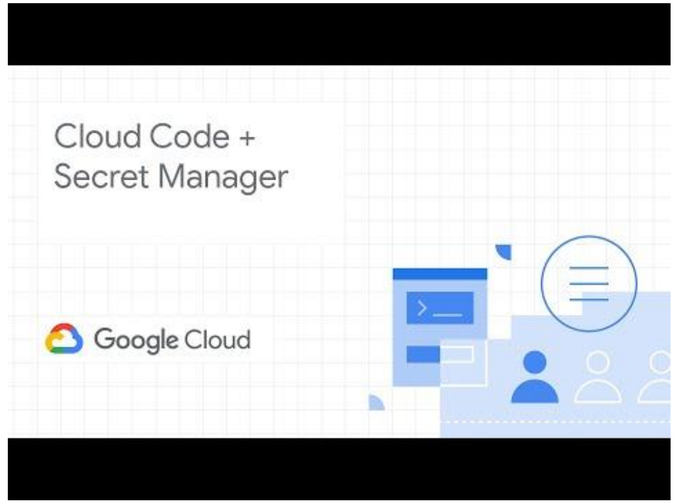
Don’t expose secrets in artifacts directly!

Secret Manager: Storing credentials securely

- Many applications require credentials to authenticate; for example, API keys, passwords, or certificates.
- Storing this information in a flat text file makes access easy but requires file protection.
- Secret Manager provides a secure, convenient way to store sensitive information.

Secret Manager features

- Global names and replication
- Versioning
- Follows principles of least privilege
- Audit logging
- Strong encryption
- Can be used in hybrid environments
- Integration with KMS



Access control using Cloud IAM

- By default, only project owners can create and access secrets within their project.
- Use Cloud IAM to grant roles and permissions at the level of the Google Cloud organization, folder, project, or secret.

Working with Cloud IAM roles

- Cloud IAM roles:
  - secretmanager.admin: Can view, edit, and access a secret.
  - secretmanager.secretAccessor: Can only access secret data.
  - secretmanager.viewer: Can view a secret's metadata and its versions, but can't edit or access secret data.
- Always apply permissions at the lowest level in the resource hierarchy.

Admin Activity Access audit logs

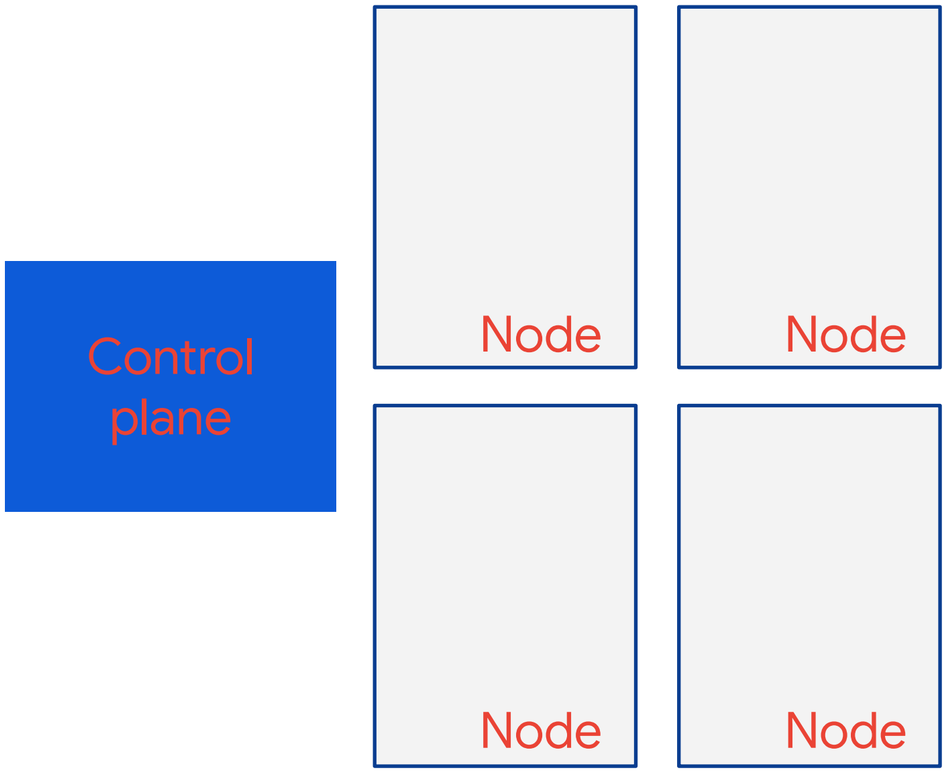
- Admin Activity Access audit logs cannot be disabled.
- All activity that creates or changes secret data is logged here.
- Setting or getting IAM policy information about secrets is logged.

Data Access audit logs

- Data Access audit logs are disabled by default.
- If logs are enabled, all access to the secret data is logged.

K8s Secrets

- Contain sensitive data, such as passwords, OAuth tokens, and SSH keys.
- Can be encrypted.
- Are used by pods to gain access to areas where they need to accomplish tasks.
- Are maintained separately from Google Cloud secrets.

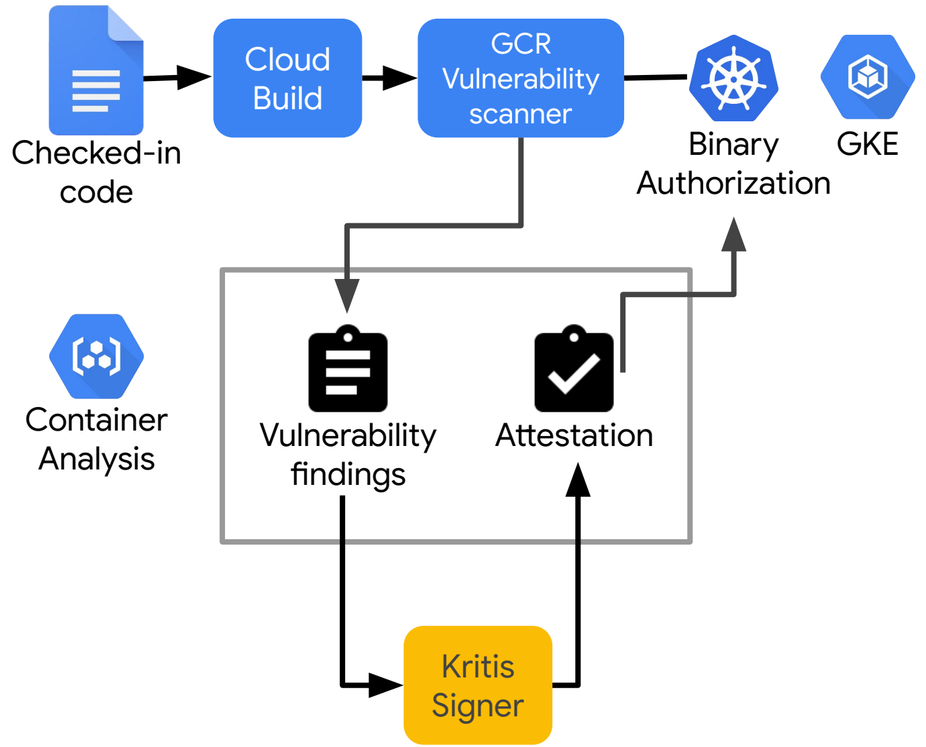


Container analysis

- Vulnerability scanning and metadata storage for containers
- Enabled at project level
- Support for Linux containers only
- Containers can be stored on either Container Registry or Artifact Registry
- Automatic scanning triggered when image pushed to registry
- Can provide notifications via Pub/Sub for integration with other services

Binary authorization allows you to enforce deploying only trusted containers into GKE

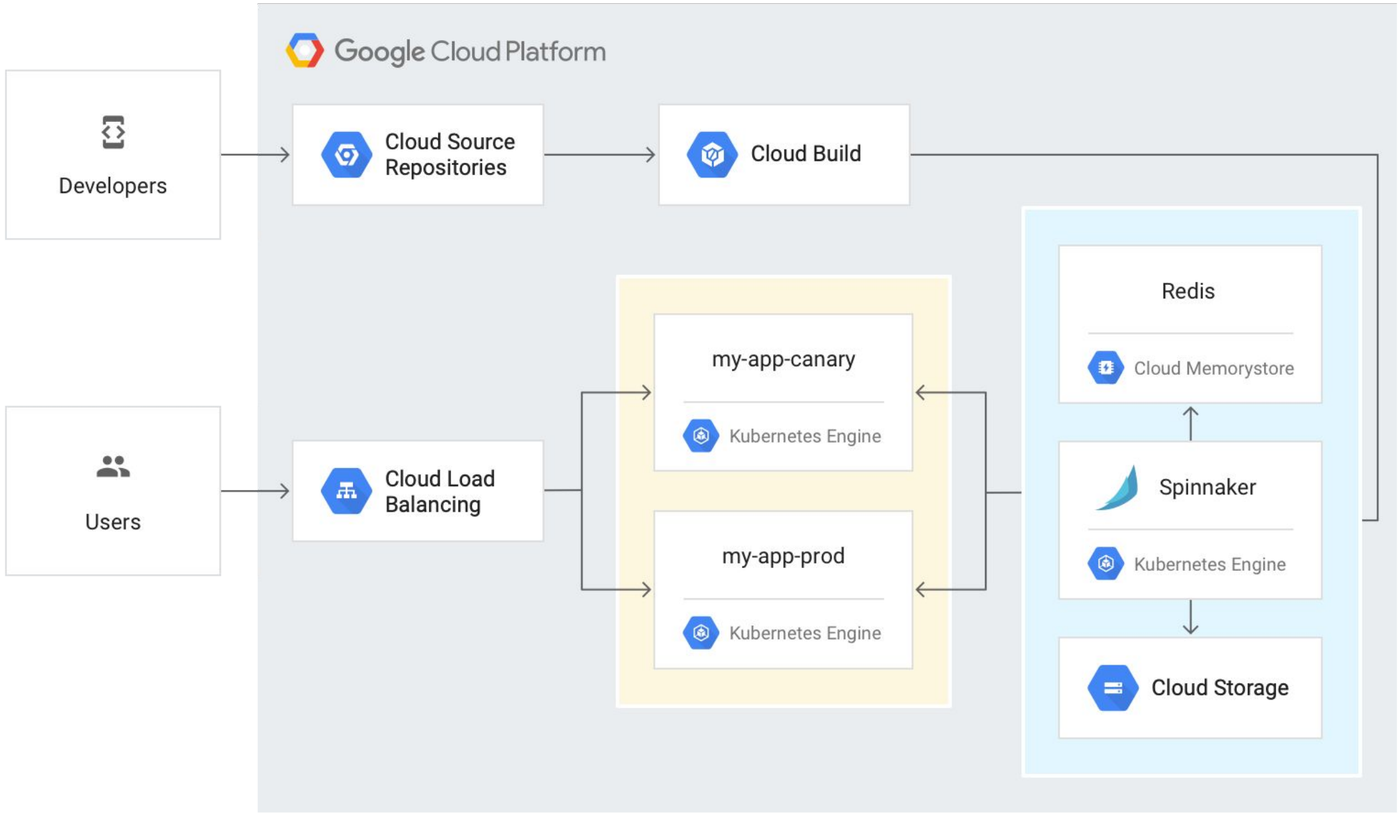
- Enable binary authorization on GKE cluster.
- Add a policy that requires signed images.
- When an image is built by Cloud Build an “attestor” verifies that it was from a trusted repository (Source Repositories, for example).
- Container Registry includes a vulnerability scanner that scans containers.



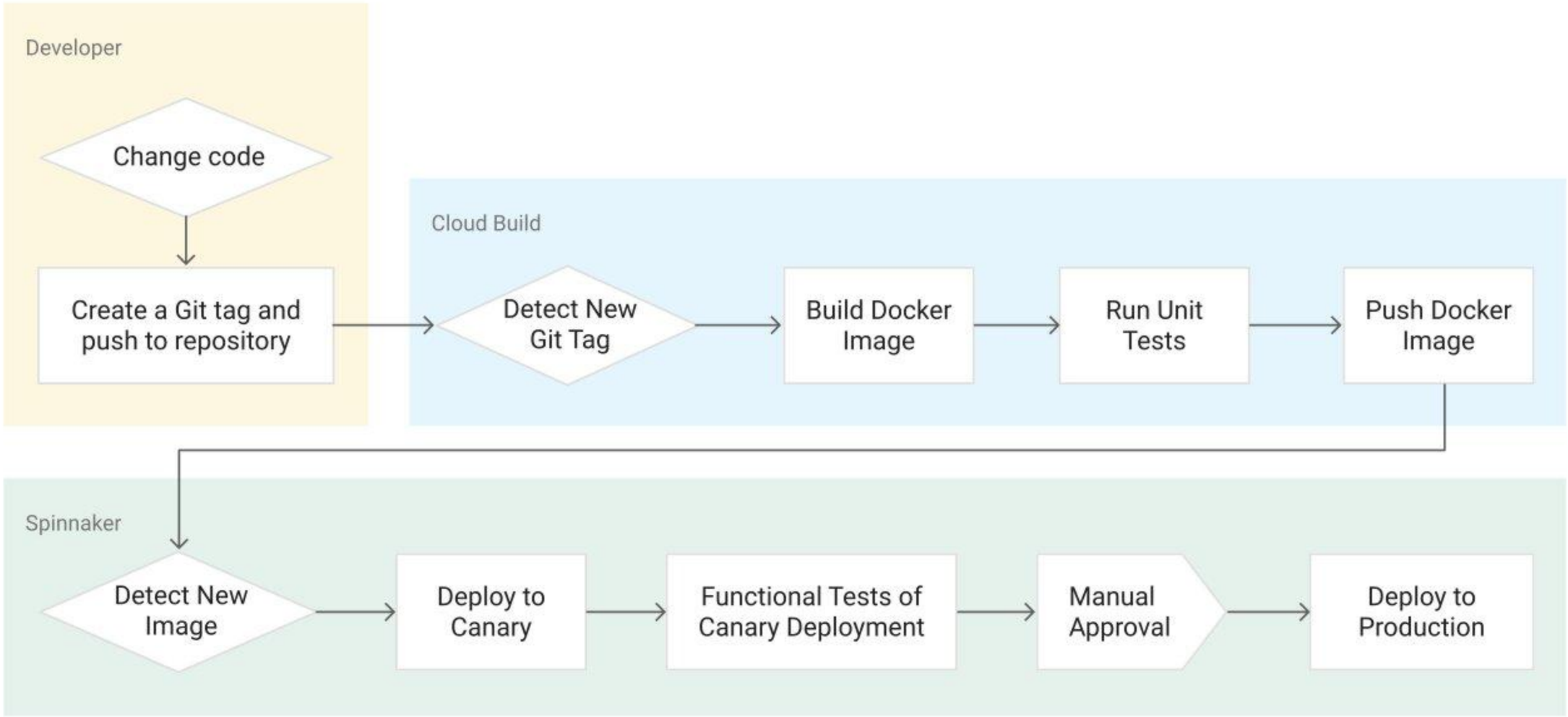
What is Spinnaker?

- Open source, multi-cloud continuous delivery tool
- Arguably Google's preferred cloud-native tool for CD
- Runs on GKE, mostly used to manage K8s applications
- Manages GKE components directly, e.g no need for administrators to update YAML config files manually. Spinnaker creates/updates all the K8s objects, such as Deployments, Replica Sets, Services etc
- Rich feature set e.g. support for canarying, blue/green, roll backs etc

CD Architecture with Spinnaker



Spinnaker CD Workflow - a closer look



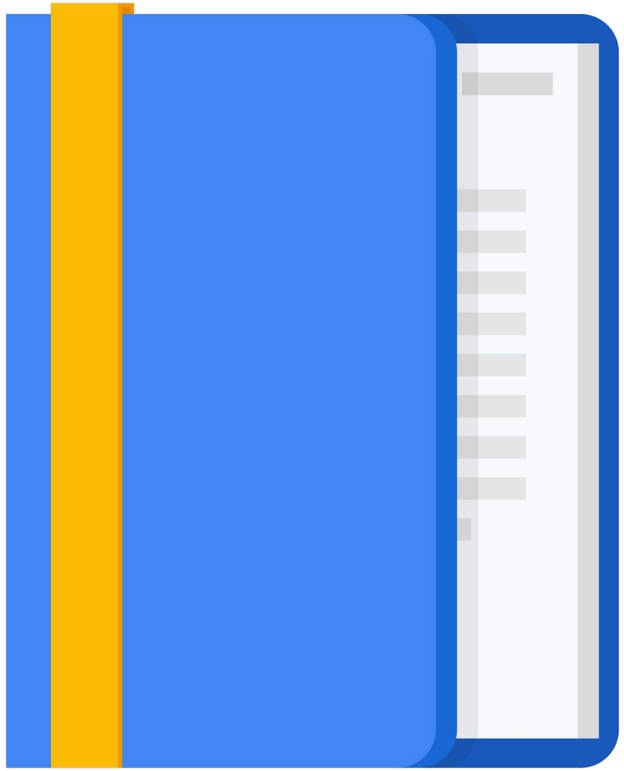
Spinnaker Demo - play the video link in the chat window on your own device

https://youtu.be/DQIu373gSKU?t=43

Agenda

Continuous Integration Pipelines

Infrastructure as Code



Moving to the cloud requires a mindset change

On-Premises	Cloud
Buy machines.	Rent machines.
Keep machines running for years.	Turn machines off as soon as possible.
Prefer fewer big machines.	Prefer lots of small machines.
Machines are capital expenditures.	Machines are monthly expenses.

In the cloud, all infrastructure needs to be disposable

- Don’t fix broken machines.
- Don’t install patches.
- Don’t upgrade machines.
- If you need to fix a machine, delete it and re-create a new one.

- To make infrastructure disposable, automate everything with code:
  - Can automate using scripts.
  - Can use declarative tools to define infrastructure.

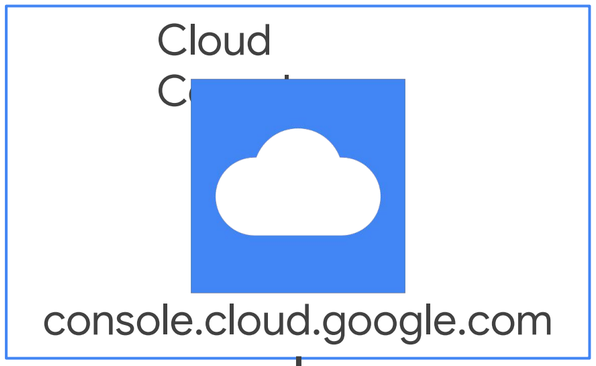
Infrastructure as code (IaC) allows for the quick provisioning and removing of infrastructures

- Build an infrastructure when needed.
- Destroy the infrastructure when not in use.
- Create identical infrastructures for dev, test, and prod.
- Can be part of a CI/CD pipeline.
- Templates are the building blocks for disaster recovery procedures.
- Manage resource dependencies and complexity.

- Google Cloud supports many IaC tools.

Terraform

Google Cloud Console, Cloud SDK, and Cloud Shell



Name	Zone	Internal IP	External IP	Connect
nginxstack-1	us-central1-f	10.128.0.3 (nic0)	35.238.84.245	SSH
nginxstack-2	us-central1-f	10.128.0.4 (nic0)	35.225.177.18	SSH
nginxstack-3	us-central1-f	10.128.0.2 (nic0)	35.239.250.238	SSH

Cloud Shell

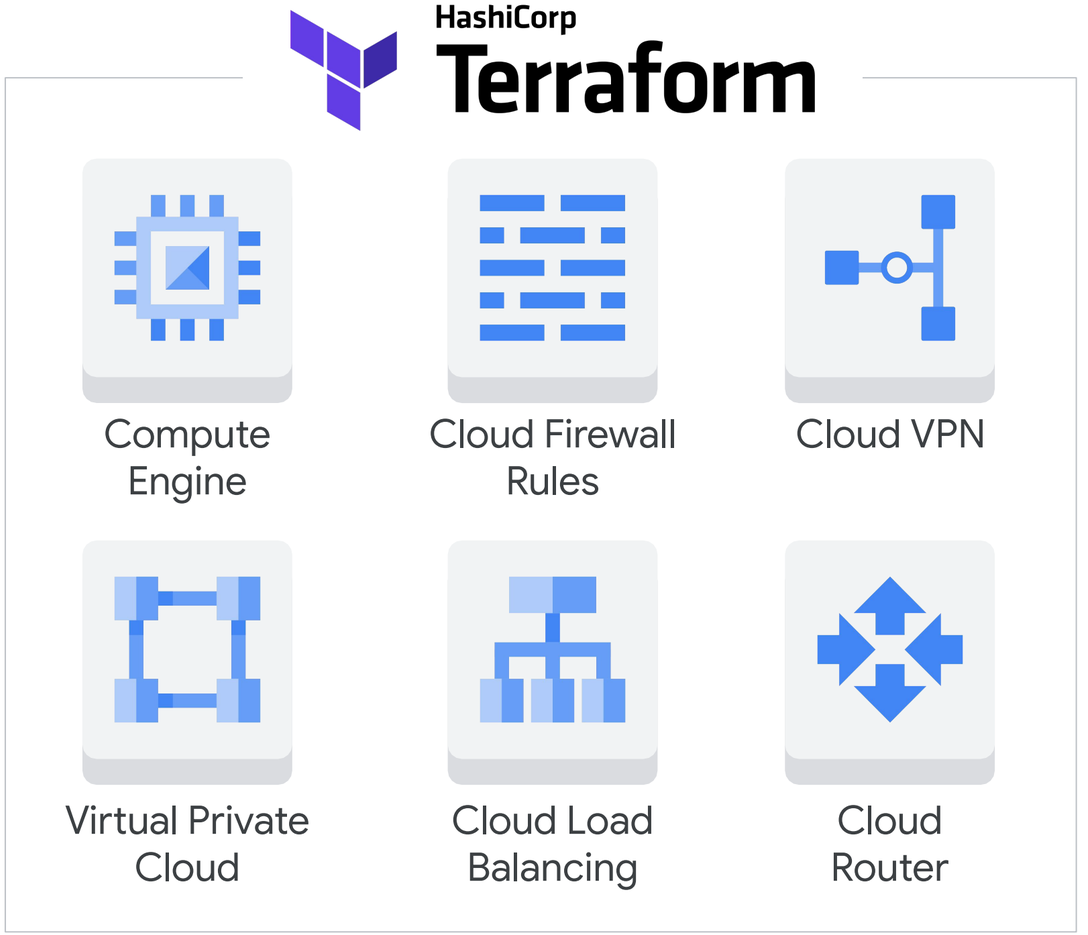
Google Cloud SDK

$ gcloud compute instances list

NAME	ZONE	INTERNAL_IP	EXTERNAL_IP
nginxstack-1	us-central1-f	10.128.0.3	35.238.84.245

Terraform is an infrastructure automation tool

- Repeatable deployment process
- Declarative language
- Focus on the application
- Parallel deployment
- Template-driven



Terraform language

- Terraform language is the interface to declare resources.
- Resources are infrastructure objects.
- The configuration file guides the management of the resource.

```
resource "google_compute_network" "default" {
  name = "${var.network_name}"
  auto_create_subnetworks = false
}

<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>" {
  # Block body
  <IDENTIFIER> = <EXPRESSION> # Argument
}
```

Terraform can be used on multiple public and private clouds

- Considered a first-class tool in Google Cloud
- Already installed in Cloud Shell

```
provider “google” {
   region= “us-central1”
}

resource “google_compute_instance” {
   name  = “instance name”
   machine_type= “n1-standard-1”
   zone  = “us-central1-f”

   disk {
      image = “image to build instance”
   }
}

output “instance_ip” {
   value = “${google_compute.ip_address}”
}
```

Example: Auto mode network with HTTP firewall rule

Auto mode network

HTTP firewall rule

Example: Auto mode network with HTTP firewall rule

```
main.tf
provider "google" {
}
```

Example: Auto mode network with HTTP firewall rule

```
main.tf
provider "google" {
}
# [START vpc_auto_create]
resource "google_compute_network" "vpc_network" {
  project  = var.project_id # Replace this with your project ID in quotes
  name                    = "my-auto-mode-network"
  auto_create_subnetworks = true

  mtu                     = 1460
}
```

Example: Auto mode network with HTTP firewall rule

```
main.tf
provider "google" {
}
.
.
.


# [START vpc_firewall_create]
resource "google_compute_firewall" "rules" {




  project     = var.project_id # Replace this with your project ID in quotes
  name        = "my-firewall-rule"
  network     = "vpc_network"
    allow {
    protocol = "tcp"
    ports  = ["80", "8080"]
  }
}
```

Deploying infrastructure with Terraform

```
terraform init
terraform plan

terraform apply
```


Google Cloud