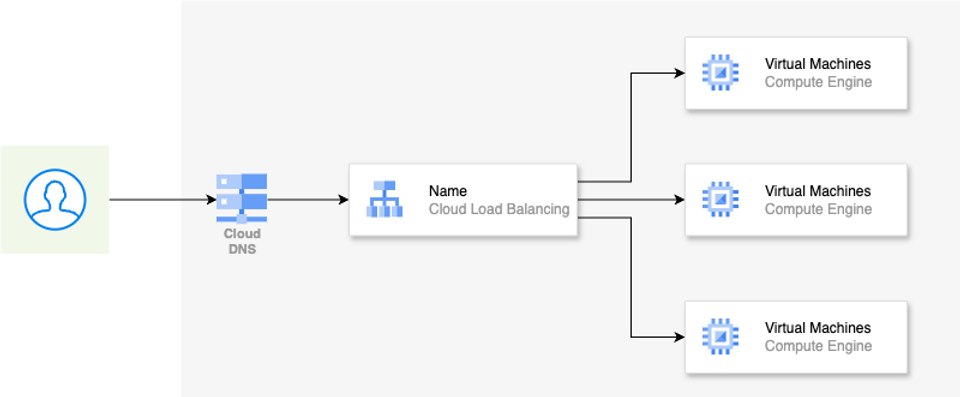
**Project 1: Web Server with Load Balancer and DNS**

**Links I used:**

[**https://www.youtube.com/watch?v=361bfIvXMBI&t=164s**](https://www.youtube.com/watch?v=361bfIvXMBI&t=164s)

This project involves setting up a basic web server (e.g., Nginx or Apache) on a Virtual Machine (VM). You'll then create a Managed Instance Group (MIG) to easily manage and scale your VMs. To distribute traffic and ensure high availability, a Layer 7 Load Balancer will be placed in front of the MIG. Finally, you'll configure DNS to map a user-friendly domain name to your load balancer, making your web application easily accessible.



**Real World application**

E-commerce companies like Amazon use this setup to handle millions of requests during peak shopping seasons. By distributing traffic across multiple web servers with a load balancer, they ensure their website remains responsive and avoids crashes even with massive traffic spikes. DNS ensures that customers can easily access the website using a memorable domain name.

**What you will learn**

By completing this project, you'll gain understanding of the below:

* **Fundamental understanding of web server deployment:**  Learn how web servers work and how to configure them.
* **Introduction to cloud computing concepts:** Gain experience with VMs, load balancing, and DNS within a cloud environment (e.g., Google Cloud, AWS, Azure).
* **Hands-on with networking:** Configure network settings for your VM and load balancer.
* **Scalability and high availability:** Understand how to scale your web application using MIGs and load balancing.

**How to get started**

* **Choose a Cloud Provider:** Select a cloud platform like Google Cloud Platform (GCP), Amazon Web Services (AWS), or Microsoft Azure and create an account. Each platform offers free tier options for experimentation.
  + GCP: <https://cloud.google.com/free>
  + AWS: <https://aws.amazon.com/free/>
  + Azure: <https://azure.microsoft.com/en-us/free/>

Call me biased but GCP is the best out there when you are starting out with your DevOps or Cloud journey.

* **VM Tutorials:** Follow tutorials specific to your chosen provider on creating and configuring VMs.
  + GCP: <https://cloud.google.com/compute/docs/quickstart-linux>
* **Web Server Guides:**  Find guides on installing and configuring web servers like Nginx or Apache.
  + Nginx: <https://www.nginx.com/resources/wiki/start/topics/tutorials/install/>
  + Apache: <https://httpd.apache.org/docs/2.4/getting-started.html>
* **Load Balancing Documentation:** Explore the load balancing services offered by your cloud provider.
  + GCP: <https://cloud.google.com/load-balancing/docs/>

**Project 2: Infrastructure as Code with Terraform**

Building on the previous project, you'll now learn how to manage your infrastructure efficiently using Terraform. This powerful Infrastructure as Code (IaC) tool allows you to define and provision your cloud resources in a declarative manner. You'll write Terraform configuration files to describe the web server, MIG, Load Balancer, and DNS you created earlier. By using Terraform, you can automate the entire infrastructure setup process, ensuring consistency, reducing manual errors, and enabling version control for your infrastructure.

You will use the basic terraform commands like init, plan and apply in this exercise.

**Real world application**

Terraform is widely used in the industry to manage infrastructure in a scalable and reliable way. Netflix uses Terraform to manage its vast and complex infrastructure, enabling them to deploy and update services quickly and reliably. By defining infrastructure as code, they can easily replicate environments, track changes, and ensure consistency across their global deployments. This helps them deliver a seamless streaming experience to millions of subscribers worldwide.

**What you will learn**

This project equips you with essential IaC skills, which are highly valued in modern DevOps practices, and teaches you how to manage infrastructure in a repeatable and scalable way.

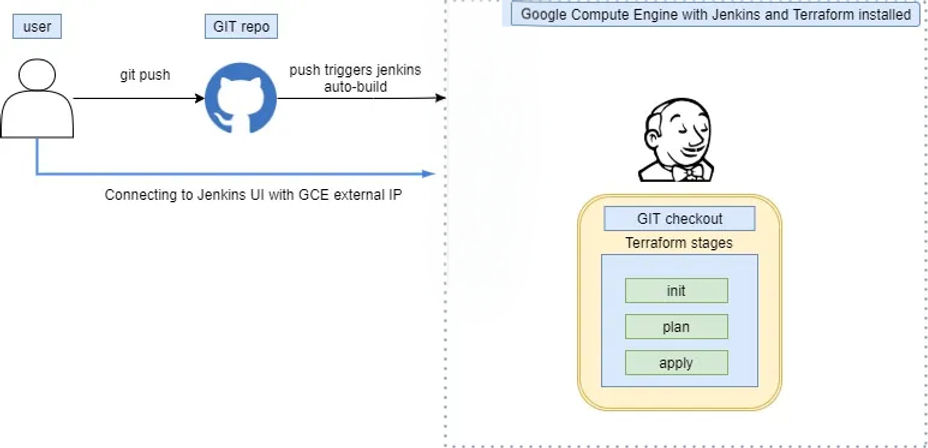
* **Infrastructure as Code (IaC):**  Learn how to define and manage infrastructure using code, making it repeatable and version-controlled.
* **Automation:** Automate the creation and management of cloud resources.
* **Reduced errors:** Minimize manual configuration and reduce the risk of human error.

**How to Get Started:**

* **Install Terraform:** Download and install Terraform on your local machine: [https://www.terraform.io/downloads.html](https://www.terraform.io/downloads.html/)
* **Terraform Documentation:** Explore the comprehensive Terraform documentation: <https://developer.hashicorp.com/terraform/docs>
* **Terraform Providers:** Learn about Terraform providers for your chosen cloud platform (e.g., Google, AWS, Azure). <https://developer.hashicorp.com/terraform/language/providers>
* **Hands-on Tutorials:** Follow tutorials that guide you through writing Terraform configurations for common cloud resources.
  + Terraform Tutorial for Google Cloud: <https://developer.hashicorp.com/terraform/tutorials/gcp-get-started>

**Project 3: CI/CD with Jenkins and Terraform**

Taking automation to the next level, this project introduces you to Continuous Integration and Continuous Delivery (CI/CD) using Jenkins. You'll create a Jenkins pipeline that automatically applies your Terraform code whenever changes are pushed to a Git repository. This means that any updates or modifications to your infrastructure defined in Terraform will be automatically deployed to your cloud environment. You'll gain experience with Git for version control and collaboration, as well as learn how to build a CI/CD pipeline that ensures your infrastructure remains up-to-date and deployments are handled in a controlled and reliable manner.



**Real world application**

Spotify relies on CI/CD pipelines with Jenkins to automate the deployment of new features and updates to its music streaming platform. This allows them to release code changes frequently and efficiently, ensuring a smooth and continuous delivery of new functionalities to their users.

**What you will learn**

This project provides a practical understanding of CI/CD principles and how they can be applied to infrastructure management, a key skill for any DevOps engineer.

* **CI/CD principles:** Understand the core concepts of CI/CD and how to implement them.
* **Automation:** Automate the deployment of your infrastructure changes.
* **Version control:** Learn how to use Git for managing your Terraform code.
* **Collaboration:**  Improve collaboration among team members by using a shared Git repository.

**How to Get Started:**

* **Install Jenkins:** Set up a Jenkins server (locally or in the cloud): <https://www.jenkins.io/doc/book/installing/>
* **Jenkins Pipeline Tutorial:** Learn how to create Jenkins pipelines: <https://www.jenkins.io/doc/book/pipeline/>
* **Git Basics:** Familiarize yourself with Git for version control: <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>
* **Integrate Terraform with Jenkins:**  Find tutorials on integrating Terraform with Jenkins for automated infrastructure deployments.
  + Example: <https://spacelift.io/blog/terraform-jenkins>

**Project 4: Docker and Kubernetes**

This project delves into the world of containerization and container orchestration. You'll start by creating a Docker image for a simple web application. Docker allows you to package your application and its dependencies into a portable and self-contained unit, ensuring consistency across different environments. Next, you'll deploy this Docker image to a Kubernetes cluster. Kubernetes is a powerful platform for managing containerized applications, providing features like automated deployment, scaling, and self-healing. This project introduces you to the core concepts of containerization and Kubernetes, essential technologies for modern cloud-native application development.

**Real world application**

Airbnb uses Docker and Kubernetes to package and deploy its microservices-based application. This allows them to scale individual services independently, ensuring optimal resource utilization and high availability. Containerization with Docker provides consistency across different environments, while Kubernetes orchestrates the deployment and management of these containers, enabling them to handle millions of bookings efficiently.

**What you will learn**

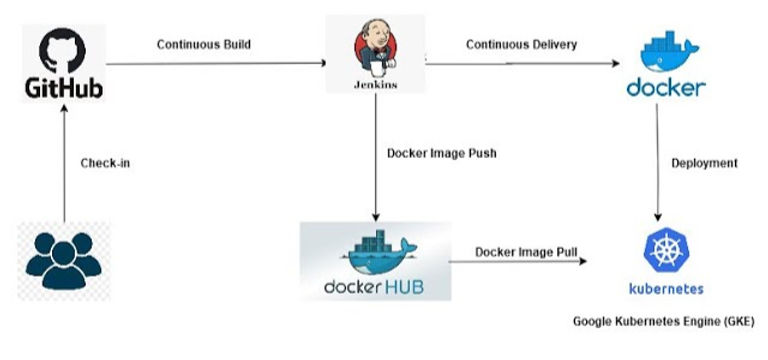
* **Containerization:** Learn how to package applications and their dependencies into Docker images for portability and consistency.
* **Kubernetes:** Gain experience with Kubernetes, a leading container orchestration platform.
* **Microservices:** Understand how to deploy and manage containerized applications, a key concept in microservices architecture.

**How to Get Started:**

* **Install Docker:** Download and install Docker Desktop on your local machine: <https://www.docker.com/products/docker-desktop/>
* **Docker Documentation:**  Get familiar with Docker concepts and commands: <https://docs.docker.com/get-started/>
* **Build a Docker Image:** Learn how to create a Dockerfile and build an image for your web application.
  + Dockerfile Tutorial: <https://docs.docker.com/engine/reference/builder/>
* **Set up Kubernetes:**  Choose a way to run Kubernetes. You can use a managed Kubernetes service like Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS), or Azure Kubernetes Service (AKS), or set up a local cluster using tools like Minikube or kind.
  + Minikube: <https://minikube.sigs.k8s.io/docs/start/>
  + kind: <https://kind.sigs.k8s.io/docs/user/quick-start/>
* **Kubernetes Basics:**  Understand basic Kubernetes concepts like pods, deployments, and services: <https://kubernetes.io/docs/concepts/overview/>

**Project 5: CI/CD for Kubernetes Deployments**

Expanding on your Kubernetes knowledge, you'll now implement a CI/CD pipeline to automate the deployment of your containerized application. Whenever your application code changes, the pipeline will automatically build a new Docker image, push it to a container registry, and deploy the updated image to your Kubernetes cluster. This streamlines the release process, allowing you to deliver new features and updates to users more quickly and reliably. This project solidifies your understanding of CI/CD principles in the context of Kubernetes deployments and demonstrates a complete workflow for managing applications in a dynamic containerized environment.



**Real-world application**

Pinterest leverages CI/CD pipelines to automate the deployment of its image-sharing platform on Kubernetes. This ensures that new features and bug fixes are rolled out quickly and reliably to their users. Automated testing and deployments help them maintain a high-quality user experience while continuously innovating.

**What you will learn**

* **Automated deployments:** Streamline the deployment process for your Kubernetes applications.
* **Faster releases:**  Deliver new features and updates to users more quickly.
* **Improved reliability:** Reduce the risk of errors during deployments.

**How to Get Started:**

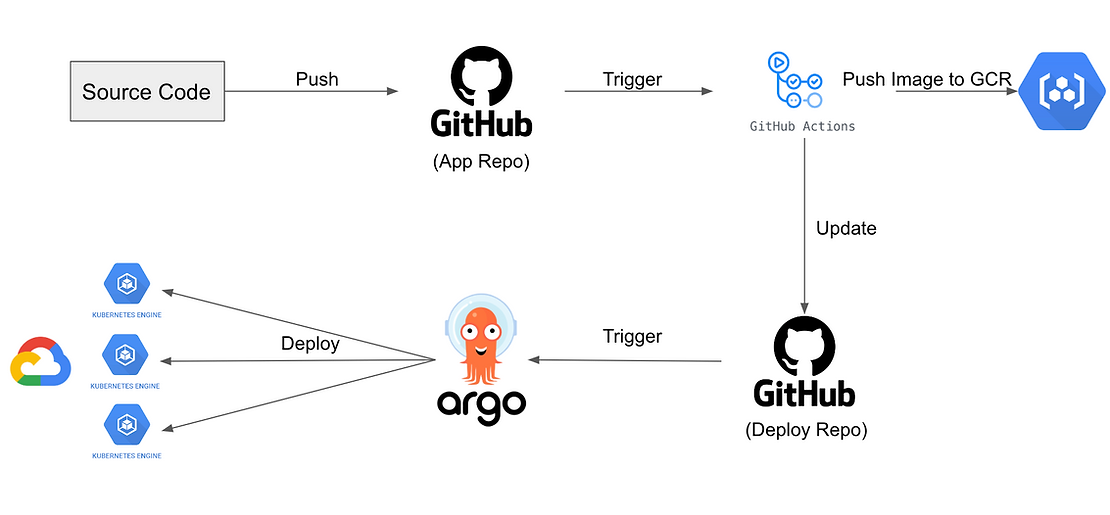
* **Choose a CI/CD Tool:** Select a CI/CD tool like Jenkins, GitLab CI/CD or CircleCI.
* **Kubernetes CI/CD Tutorials:**  Find tutorials specific to your chosen CI/CD tool and cloud provider on how to set up pipelines for Kubernetes deployments.
  + Jenkins and Kubernetes: [invalid URL removed]
  + GitLab CI/CD and Kubernetes: [invalid URL removed] [invalid URL removed]
* **Container Registry:** Choose a container registry to store your Docker images (e.g., Docker Hub, Google Container Registry, Amazon ECR).
* **Kubernetes Deployment Configurations:** Learn how to define Kubernetes deployments and services using YAML files.

**Project 6: GitOps for Kubernetes Deployments**

This project introduces you to GitOps, a modern approach to managing Kubernetes deployments using Git as the single source of truth. In a GitOps workflow, any changes to your application or its configuration are made through Git commits. These changes trigger automated processes that update your Kubernetes cluster to match the desired state defined in your Git repository. This approach enhances collaboration, simplifies rollbacks, and improves the auditability of your deployments. I have made an indepth ArgoCD tutorial to make you an expert in ArgoCD. You can checkout the tutorial videos here: <https://www.youtube.com/playlist?list=PLsgfdeFHc1WBgiWDOFS8S91ElJQxi_Qpc>

If you just interested in implementing the CI/CD pipeline using ArgoCD, you can refer this video: <https://youtu.be/OH2D0cqS0mw>

**Architecture Diagram:**



**What you will learn:**

* **Increased Automation:** Automate deployments, configuration management, and monitoring through Git.
* **Improved Collaboration:**  Use familiar Git workflows for collaboration and code reviews.
* **Enhanced Stability:**  Ensure consistency between your desired state and the actual state of your cluster.
* **Easier Rollbacks:**  Quickly revert to previous application versions by simply reverting Git commits.
* **Auditing and Traceability:** Track all changes to your Kubernetes environment through Git history.

**Real-World Application:**

GitOps is becoming increasingly popular for managing Kubernetes deployments in production environments. It helps teams streamline deployments, improve reliability, and reduce the risk of errors.

Companies like Weaveworks, who pioneered GitOps, use this approach to manage the deployments of their own products and services on Kubernetes. This allows them to maintain a declarative and auditable deployment process, ensuring that their infrastructure is always in sync with the desired state defined in Git.

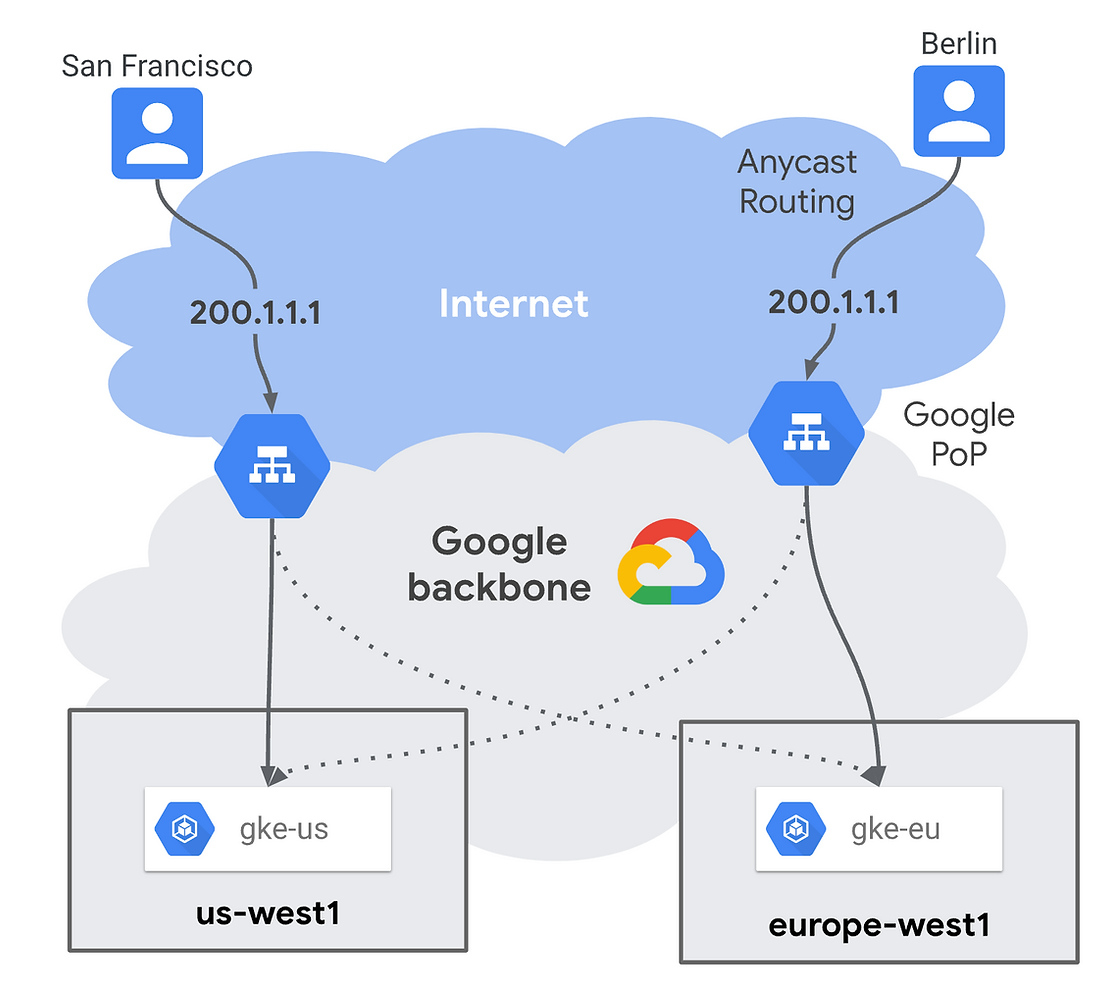
**How to Get Started:**

* **Choose a GitOps Tool:** Select a GitOps tool like Argo CD or Flux.
  + Argo CD: <https://argo-cd.readthedocs.io/en/stable/>
  + Flux: <https://fluxcd.io/>
* **GitOps Tutorials:** Find tutorials on how to set up and use your chosen GitOps tool with Kubernetes.
  + Argo CD Tutorial: <https://argo-cd.readthedocs.io/en/stable/getting_started/>
* **Kubernetes Manifests:** Learn how to define your Kubernetes deployments, services, and other resources using YAML manifests.
* **Git Workflow:**  Ensure you have a good understanding of Git branching, merging, and pull requests.

This new GitOps project adds another valuable dimension to your learning path, demonstrating a modern and efficient way to manage Kubernetes deployments.

**Project 7: Multi-Cluster Deployment with Global Load Balancing**

This advanced project challenges you to deploy your application across multiple Kubernetes clusters located in different regions. You'll then use a Global Load Balancer to distribute traffic across these clusters based on factors like user location and server load. This approach ensures high availability, fault tolerance, and optimal performance for users around the world. You'll gain valuable experience in designing and managing globally distributed applications, a crucial skill for large-scale deployments. This project combines your knowledge of Kubernetes, networking, and system design to tackle real-world challenges in application scalability and resilience, preparing you for complex, real-world cloud deployments.



**Real world application**

This project simulates a complex, real-world scenario where you need to deploy a highly available and scalable application across multiple regions.

Global companies like Google use multi-cluster deployments with global load balancing to serve their applications to users worldwide with low latency. By distributing their application across multiple regions and using a global load balancer, they ensure high availability and optimal performance for users regardless of their location.

**What you will learn**

* **Global scale:** Learn how to deploy applications at a global scale.
* **High availability:** Ensure your application is always available, even in case of regional outages.
* **Improved performance:** Reduce latency for users by serving them from the closest cluster.
* **Disaster recovery:**  Provide a robust disaster recovery solution by distributing your application across multiple regions.

**How to Get Started:**

* **Multi-Cluster Kubernetes:** Research how to set up and manage multiple Kubernetes clusters in different regions on your chosen cloud platform.
  + <https://cloud.google.com/kubernetes-engine/docs/concepts/multi-cluster-ingress>
  + <https://cloud.google.com/kubernetes-engine/docs/how-to/multi-cluster-ingress-setup>
* **Global Load Balancing:** Explore the global load balancing solutions offered by your cloud provider.
* **Networking Concepts:**  Strengthen your understanding of networking concepts like DNS, routing, and load balancing.