High-Level and Low-Level Design Document

Project: Deploying a Prefect Worker on AWS ECS Fargate using Terraform

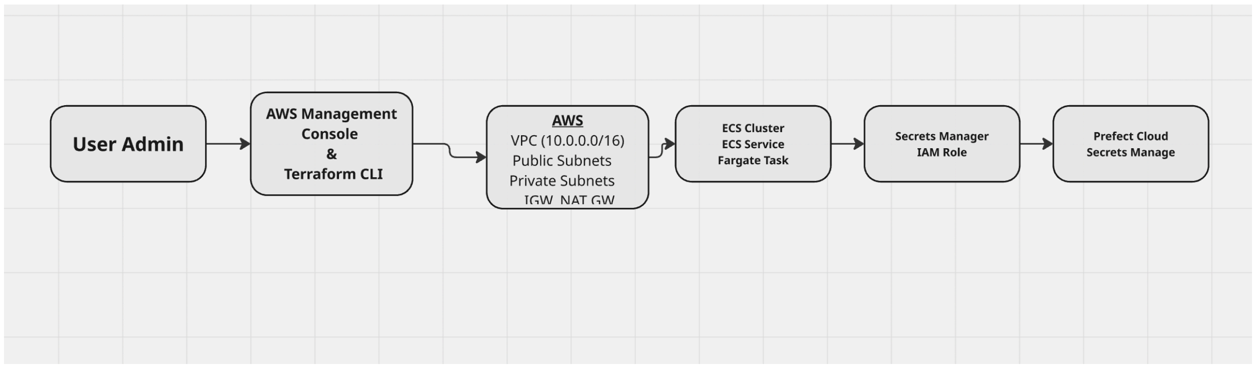
High-Level Design (HLD)

Objective

Deploy a Prefect 2.0 worker container on AWS ECS Fargate using Infrastructure as Code (Terraform), integrated securely with Prefect Cloud and a custom AWS environment.

Components Overview

- **VPC and Networking**
  - VPC with CIDR block `10.0.0.0/16`
  - Three public and three private subnets across multiple Availability Zones
  - Internet Gateway attached for public subnet internet access
  - NAT Gateway provisioned for private subnet internet access
  - Route tables configured for public and private subnets
- **Compute (ECS)**
  - ECS Cluster named `prefect-cluster`
  - ECS Fargate Task Definition with:
    - Image: `prefecthq/prefect:2-latest`
    - Resource limits defined (CPU, Memory)
  - ECS Fargate Service to manage tasks
- **Security**
  - IAM Role: `prefect-task-execution-role`
    - Attached policies:
      - `AmazonECSTaskExecutionRolePolicy`
      - Custom inline policy for Secrets Manager access
  - AWS Secrets Manager:
    - Secret named `prefect-api-key`
- **Service Discovery**
  - AWS Cloud Map Private DNS namespace `default.prefect.local` for internal ECS service communication
- **Prefect Cloud Integration**
  - Prefect API Key securely injected into the task environment
  - Worker connects to Prefect Cloud and registers with `ecs-work-pool`



Low-Level Design (LLD)

1. VPC and Subnetting

- **VPC**:
  - Name: `prefect-ecs-vpc`
  - CIDR Block: `10.0.0.0/16`
- **Public Subnets**:
  - Three subnets (one per AZ)
  - Auto-assign public IP enabled
- **Private Subnets**:
  - Three subnets (one per AZ)
  - No auto-assign public IP
- **Internet Gateway**:
  - Attached to VPC for public subnet access
- **NAT Gateway**:
  - Created in a public subnet for private subnet internet access
- **Route Tables**:
  - Public Route Table routes `0.0.0.0/0` to Internet Gateway
  - Private Route Table routes `0.0.0.0/0` to NAT Gateway

---

2. IAM Roles

- **Task Execution Role (prefect-task-execution-role)**:
  - Trust Policy: Allows ECS Tasks to assume role
  - Attached Policies:
    - `AmazonECSTaskExecutionRolePolicy`
    - Custom policy for reading Prefect secrets from Secrets Manager

---

3. ECS Cluster and Services

- **ECS Cluster**:
  - Name: `prefect-cluster`
- **Service Discovery**:
  - Created Private DNS namespace `default.prefect.local`
- **Task Definition**:
  - Container Image: `prefecthq/prefect:2-latest`
  - CPU and Memory allocations
  - Environment Variables:
    - PREFECT_API_KEY
    - PREFECT_ACCOUNT_ID
    - PREFECT_WORKSPACE_ID
    - PREFECT_API_URL
- **ECS Service**:
  - Launch type: Fargate
  - Subnets: Private Subnets
  - Security Groups: Allow only necessary communication (optional if needed)

---

4. Secrets Management

- **AWS Secrets Manager**:
  - Secret Name: `prefect-api-key`
  - Stores the Prefect API key securely
  - Retrieved dynamically in task environment variables

---

5. Prefect Cloud Configuration

- **Work Pool Name**: `ecs-work-pool`
- **Worker Name**: `dev-worker`
- Worker automatically registers itself with Prefect Cloud after ECS deployment.

Future Improvements

- Implement auto-scaling based on task queue size or CPU/Memory usage
- Set up CloudWatch log groups for task and service monitoring
- Introduce Terraform modules for better scalability and code reuse
- Add Application Load Balancer (ALB) if future services need external access
- Automate deployment via CI/CD pipeline (GitHub Actions, CircleCI)