Got it! Here's how we'll tackle your assignment:

**Solution Breakdown**

**1. Architecture Overview**

We will deploy a **Node.js API** on **AWS ECS (Fargate)** for serverless, scalable container management. The API will be secured and rate-limited at the infrastructure level. The architecture includes:

* **AWS API Gateway** → For rate limiting and authentication.
* **AWS Cognito** → To manage API authentication.
* **AWS ECS (Fargate)** → To run the API in containers.
* **AWS ALB (Application Load Balancer)** → To handle blue-green deployments.
* **AWS Route 53** → For DNS management.
* **AWS RDS (PostgreSQL)** → To store API usage data if needed.
* **AWS CloudWatch** → For monitoring and logging.

**2. Blue-Green Deployment**

We will deploy two separate ECS services (Blue & Green) behind an ALB. Traffic switching will be handled via **Route 53 and ALB Target Groups**, ensuring zero downtime.

**3. CI/CD Pipeline**

* **GitHub Actions/GitLab CI/CD** will automate:
  + Code linting & formatting (ESLint, Prettier)
  + Code quality checks (SonarQube)
  + SAST (GitHub CodeQL or Snyk)
  + Container build & push to AWS ECR
  + Terraform infrastructure provisioning
  + Deployment to AWS ECS using blue-green strategy

**4. Infrastructure as Code (Terraform)**

* **Modularized Terraform Setup:**
  + networking (VPC, Subnets, Security Groups)
  + ecs (Cluster, Services, Task Definitions)
  + alb (Load Balancer, Target Groups, Listeners)
  + api\_gateway (Rate limiting, Authentication)
  + cicd (IAM Roles, Pipelines)
  + monitoring (CloudWatch Alarms & Logs)

**5. Deliverables**

* **GitHub/GitLab Repository** with:
  + Terraform modules for infrastructure.
  + CI/CD pipeline configuration (GitHub Actions/GitLab CI).
  + Architecture diagram.

**Next Steps**

* I'll prepare the **Terraform files** and **CI/CD scripts**.
* Would you like me to generate the **architecture diagram** as well?
* Do you have a preferred Terraform structure or should I use a standard modular approach?

Let me know, and I’ll start building the solution! 🚀

**You said:**

start building the solution

**ChatGPT said:**

I'll start by setting up the **Terraform infrastructure** in a modular way. Here's the plan:

**Terraform Modules to be Created:**

1. **Networking** → VPC, Subnets, Security Groups, NAT Gateway, Route Tables
2. **ECS Cluster** → ECS Cluster, Task Definitions, Fargate Services
3. **Application Load Balancer (ALB)** → ALB, Target Groups, Listeners
4. **API Gateway & Cognito** → Rate limiting and authentication setup
5. **CI/CD Pipeline** → IAM roles, GitHub Actions/GitLab CI/CD scripts
6. **Monitoring** → CloudWatch logs, alarms, and metrics

I'll set up the **Terraform project structure** first and then move on to the **CI/CD pipeline**.

Let me create the Terraform files now.

Terraform Aws Ecs

I've initialized the **Terraform project structure** with modularized components for networking, ECS, ALB, API Gateway, CI/CD, and monitoring.

Next, I'll flesh out the individual Terraform modules. Let me know if you have any specific preferences! 🚀

**You said:**

any specific preferences as in?

**ChatGPT said:**

Good question! Here are a few things you might want to specify:

1. **AWS Region** – Do you have a preferred region (e.g., us-east-1, eu-west-1)?
2. **Container Runtime** – Are we sticking with Fargate (serverless) or using EC2 for ECS tasks?
3. **Database Requirement** – Do you need a database like **RDS (PostgreSQL)**, or is this a stateless API?
4. **Rate Limiting Strategy** – Should we use API Gateway’s built-in throttling, WAF, or a combination?
5. **Authentication Mechanism** – Cognito user pools, API Gateway IAM, or another method?
6. **CI/CD Pipeline** – Are we using **GitHub Actions** or **GitLab CI/CD**?
7. **Monitoring** – Do you want **CloudWatch Alarms**, AWS X-Ray, or third-party tools like Datadog?

If you don’t have strong preferences, I’ll go with **AWS best practices** for everything! Let me know. 🚀

**You said:**

AWS Region will be us-east-1 Container Runtime let it be Fargate DB we will stick with RDS Rate Limiting Strategy API Gateway's built-in throttling Authentication - Cognito CI/CD GitHub Actions Monitoring CloudWatch

**ChatGPT said:**

Memory updated

Got it! I'll now proceed with setting up the **Terraform modules** based on these preferences.

I'll start with:

1. **Networking** (VPC, subnets, security groups)
2. **ECS Fargate cluster** (with service and task definitions)
3. **Application Load Balancer (ALB) & Target Groups**
4. **API Gateway & Cognito** for authentication and rate limiting
5. **RDS** (PostgreSQL) setup
6. **CI/CD pipeline with GitHub Actions**
7. **CloudWatch for monitoring**

I’ll update the Terraform code accordingly. 🚀 Stay tuned!

**You said:**

yeah go ahead

**ChatGPT said:**

Edited

I've updated the Terraform configuration to match your preferences:

* **AWS Region**: us-east-1
* **Container Runtime**: Fargate
* **RDS (PostgreSQL)** included
* **API Gateway with Cognito for authentication**
* **CI/CD via GitHub Actions**
* **CloudWatch for monitoring**

Next, I'll define the Terraform modules for each component. Let me know if you need any tweaks! 🚀

**You said:**

okay go ahead

**ChatGPT said:**

Now, I'll define the Terraform modules inside the modules/ directory. Here’s the plan:

1. **Networking Module** (modules/networking)
   * VPC, subnets (public/private), security groups, and NAT gateway.
2. **ECS Module** (modules/ecs)
   * ECS Cluster, Fargate task definition, and service.
3. **RDS Module** (modules/rds)
   * PostgreSQL instance, subnet group, security group.
4. **ALB Module** (modules/alb)
   * ALB, target groups, listeners, and security rules.
5. **API Gateway Module** (modules/api\_gateway)
   * API Gateway configuration with Cognito authentication and rate limiting.
6. **CI/CD Module** (modules/cicd)
   * IAM roles for GitHub Actions, pipeline setup.
7. **Monitoring Module** (modules/monitoring)
   * CloudWatch alarms, log groups, and metrics.

I'll start with the **networking module** and build from there. Stay tuned! 🚀

Edited

I've enhanced the Terraform configuration by adding:

* **Networking improvements**: NAT gateway & VPC endpoints.
* **ECS Task Resources**: CPU (512) & Memory (1024).
* **RDS Enhancements**: Storage (20GB) & Backup retention (7 days).
* **ALB**: HTTPS enabled.
* **API Gateway**: Logging enabled.
* **CI/CD**: Added SAST & code quality checks.
* **Monitoring**: Alarms enabled.

Next, I'll define the **Terraform module files** inside modules/. Let me know if you need changes! 🚀

I've updated the Terraform configuration with:

* **Networking module**: Added security group rules for ingress/egress.
* **ECS module**: Defined desired count (2) and container port (8080).
* **RDS module**: Set multi-AZ for high availability and disabled public access.
* **ALB module**: Included an ACM certificate for HTTPS.
* **API Gateway**: Added prod stage.
* **CI/CD**: Enabled automatic deployment.
* **Monitoring**: Added CloudWatch metrics.

Next, I'll define the **module files** inside modules/. 🚀 Let me know if you need any changes!

Now, I'll define the **Terraform module files** for each component. Here’s the order:

1. **Networking (modules/networking/main.tf)** → VPC, subnets, security groups
2. **ECS (modules/ecs/main.tf)** → ECS cluster, task definition, service
3. **RDS (modules/rds/main.tf)** → PostgreSQL instance, security groups
4. **ALB (modules/alb/main.tf)** → ALB, target groups, listeners
5. **API Gateway (modules/api\_gateway/main.tf)** → Rate limiting, Cognito authentication
6. **CI/CD (modules/cicd/main.tf)** → GitHub Actions pipeline setup
7. **Monitoring (modules/monitoring/main.tf)** → CloudWatch alarms, logs

I'll start with the **networking module** and update the code now. 🚀

Edited

I've updated the Terraform configuration with:

* **Networking module**: Added security group rules for ingress/egress.
* **ECS module**: Defined desired count (2) and container port (8080).
* **RDS module**: Set multi-AZ for high availability and disabled public access.
* **ALB module**: Included an ACM certificate for HTTPS.
* **API Gateway**: Added prod stage.
* **CI/CD**: Enabled automatic deployment.
* **Monitoring**: Added CloudWatch metrics.

Next, I'll define the **module files** inside modules/. 🚀 Let me know if you need any changes!

**You said:**

okay

**ChatGPT said:**

Now, I'll create the **Terraform module files** for each component inside modules/. Here's what I'll define:

**1. Networking Module (modules/networking/main.tf)**

* VPC, public/private subnets, NAT Gateway, security groups.

**2. ECS Module (modules/ecs/main.tf)**

* ECS cluster, task definition, Fargate service.

**3. RDS Module (modules/rds/main.tf)**

* PostgreSQL database, subnet group, security groups.

**4. ALB Module (modules/alb/main.tf)**

* Application Load Balancer, target groups, listeners.

**5. API Gateway Module (modules/api\_gateway/main.tf)**

* API Gateway configuration with Cognito authentication and rate limiting.

**6. CI/CD Module (modules/cicd/main.tf)**

* GitHub Actions IAM role, permissions, pipeline integration.

**7. Monitoring Module (modules/monitoring/main.tf)**

* CloudWatch logs, metrics, alarms.

I'll start by adding the **networking module**. Let me know if you need any specific changes! 🚀

**You said:**

start defining the networking module

**ChatGPT said:**

Your **networking module** is already referenced in the root main.tf. Now, I'll define the networking module inside modules/networking/.

**Networking Module Files**

1. **modules/networking/main.tf** → Defines VPC, subnets, security groups.
2. **modules/networking/variables.tf** → Declares input variables.
3. **modules/networking/outputs.tf** → Exports VPC & subnet details.

This will create: ✅ **VPC**  
✅ **Public & Private Subnets**  
✅ **Security Groups**  
✅ **Internet Gateway**  
✅ **Public Route Table**

Let me know if you need changes before I move on to ECS! 🚀

2/2

**You said:**

okay lets start

**ChatGPT said:**

Great! Your **networking module** is now defined. Next, I’ll set up the **ECS module** inside modules/ecs/.

**ECS Module Plan**

This module will handle:  
✅ **ECS Cluster** (Fargate)  
✅ **Task Definition** (Node.js container)  
✅ **ECS Service** (Blue-Green Deployment)  
✅ **IAM Roles** for ECS Execution

I'll now define the Terraform files for this module. 🚀

**You said:**

okay

**ChatGPT said:**

I'll now define the **ECS module** inside modules/ecs/.

**ECS Module Files**

1. **modules/ecs/main.tf** → Defines ECS Cluster, Task Definition, Service.
2. **modules/ecs/variables.tf** → Declares input variables.
3. **modules/ecs/outputs.tf** → Exports ECS details.

**✅ What This Module Does:**

* Creates an **ECS Cluster**.
* Defines an **ECS Task** (Node.js container).
* Configures an **ECS Service** with **Blue-Green Deployment**.
* Sets up an **IAM Execution Role** for pulling container images.
* Creates a **Security Group** for the ECS service.

Next, I'll work on **RDS (PostgreSQL) module**. Let me know if you need changes! 🚀

Now, I'll define the **RDS module** inside modules/rds/.

**RDS Module Plan**

This module will handle:  
✅ **RDS Instance (PostgreSQL)**  
✅ **Subnet Group** (Private Subnets)  
✅ **Security Group** (Restricted Access)  
✅ **Multi-AZ Deployment**

**✅ What This Module Does:**

* Creates an **RDS PostgreSQL Instance** with **Multi-AZ** for high availability.
* Configures a **Subnet Group** (Runs in **Private Subnets**).
* Creates an **RDS Security Group** (Restricts access).
* Enables **Automated Backups** (7 Days Retention).

Next, I'll define the **ALB module**. Let me know if you need any changes! 🚀

Top of Form

Now, I'll define the **ALB (Application Load Balancer) module** inside modules/alb/.

**ALB Module Plan**

This module will handle:  
✅ **Application Load Balancer (ALB)**  
✅ **Target Group** for ECS Service  
✅ **HTTPS Listener with ACM Certificate**  
✅ **Security Group for ALB**

**✅ What This Module Does:**

* **Creates an ALB** that distributes traffic to ECS.
* **Redirects HTTP to HTTPS** for security.
* **Uses an ACM Certificate** for SSL/TLS encryption.
* **Creates a Target Group** for ECS.
* **Defines a Security Group** for controlled access.

Next, I'll work on the **API Gateway module**. Let me know if you need any changes! 🚀

Now, I'll define the **API Gateway module** inside modules/api\_gateway/.

**API Gateway Module Plan**

This module will handle:  
✅ **REST API Gateway**  
✅ **Cognito Authentication**  
✅ **Rate Limiting with Throttling**  
✅ **Logging and Monitoring**

**✅ What This Module Does:**

* **Creates an API Gateway** for the application.
* **Enforces authentication using Cognito**.
* **Implements rate limiting** to control API usage.
* **Integrates with ALB** to forward requests to ECS.
* **Logs API requests to CloudWatch** for monitoring.

Next, I'll set up the **CI/CD pipeline with GitHub Actions**. Let me know if you need any changes! 🚀

Now, I'll define the **CI/CD pipeline with GitHub Actions** inside modules/cicd/.

**CI/CD Pipeline Plan**

This pipeline will handle:  
✅ **Code Quality Checks** (Linting, Code Formatting)  
✅ **Static Application Security Testing (SAST)**  
✅ **Building and Pushing Docker Image to ECR**  
✅ **Deploying to AWS ECS using Blue-Green Deployment**  
✅ **Terraform Apply for Infrastructure Updates**

**GitHub Actions Workflow (.github/workflows/deploy.yml)**

**✅ What This CI/CD Pipeline Does:**

* **Runs Linting & Code Quality Checks** (Prevents bad code from merging).
* **Performs Security Scans (SAST)** using Snyk.
* **Builds & Pushes Docker Image to AWS ECR**.
* **Deploys New Task to ECS (Blue-Green Deployment)**.
* **Uses GitHub Secrets** for storing AWS credentials securely.

Next, I'll integrate **Terraform apply for infrastructure updates** into the pipeline. Let me know if you need any changes! 🚀

Now, I'll integrate **Terraform Apply** into the CI/CD pipeline to automate infrastructure updates.

**Terraform in CI/CD Pipeline**

✅ **Terraform Init, Validate & Plan** before deployment  
✅ **Terraform Apply on main branch merges**  
✅ **Ensures infrastructure is updated automatically**

Top of Form

**I'll define the Monitoring Module (modules/monitoring/) using AWS CloudWatch.**

**Monitoring Plan**

**✅ ECS Logs & Metrics in CloudWatch  
✅ API Gateway Logging enabled  
✅ CloudWatch Alarms for ECS & RDS  
✅ Dashboards for Metrics Visualization**

**✅ What This Update Adds:**

* **Terraform Init, Validate & Plan** to check infra changes.
* **Terraform Apply on main branch merges** to auto-deploy infra updates.
* **Ensures ECS, API Gateway, ALB, Cognito, and other AWS resources stay up to date**.
* **Terraform workflow runs before building & deploying the app**.

Now, I'll finalize the **Terraform Outputs & Documentation** to wrap this up. 🚀 Let me know if you need any changes!

Now, I'll define **Terraform Outputs & Documentation** to complete the solution.

Module Structure:  
  
📂 modules/

├── networking/ # VPC, Subnets, Security Groups

├── ecs/ # ECS Cluster & Service

├── rds/ # RDS PostgreSQL Database

├── alb/ # Application Load Balancer

├── api\_gateway/ # API Gateway & Cognito

├── cicd/ # CI/CD Pipeline with GitHub Actions

├── monitoring/ # CloudWatch Monitoring & Alarms

Top of Form

Top of Form

Bottom of Form