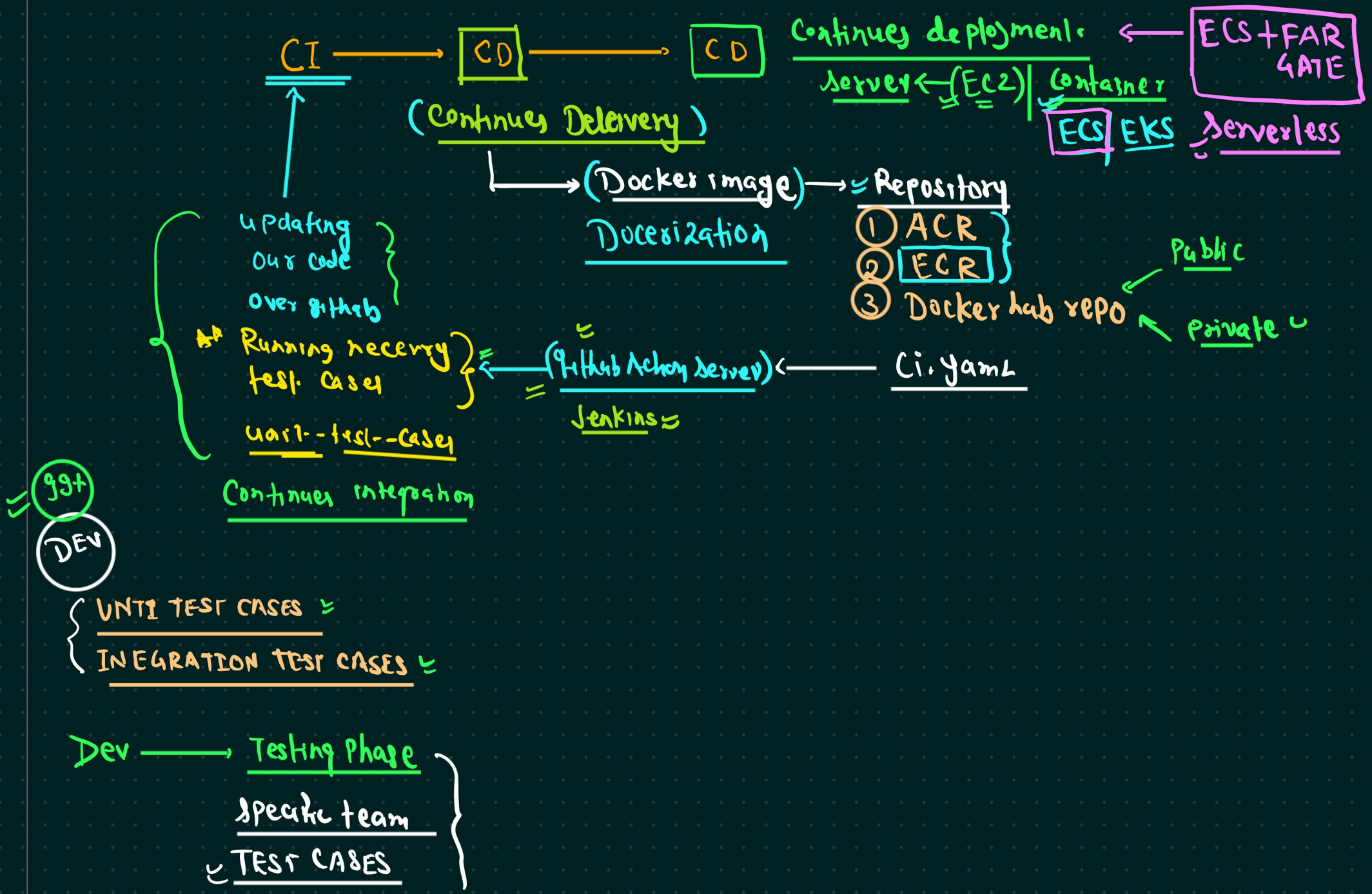
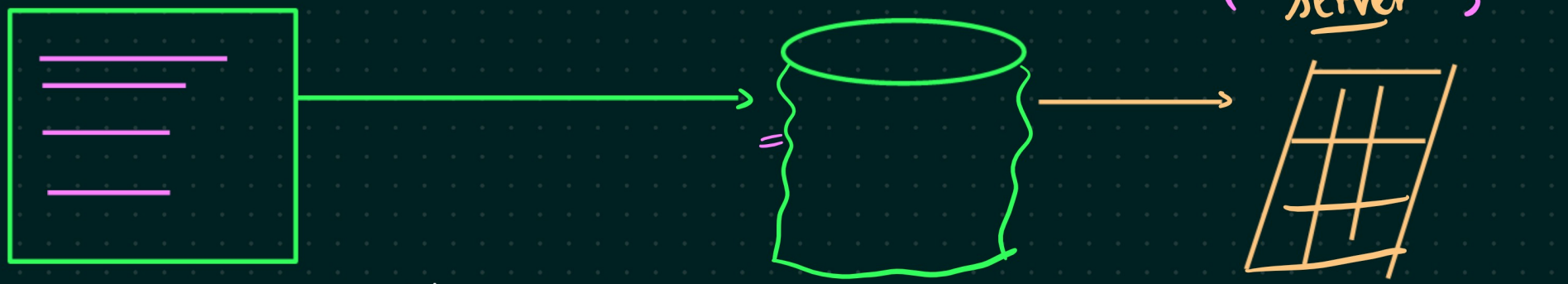

CI
CD
CD
Continues deployment
ECS+FARGATE
server
EC2
Container
ECS
EKS
Serverless
(Continues Delevery)
(Docker image)
Repository
Docerization
1 ACR
2 ECR
3 Docker hub repo
Public
Private
Updating Our code over github
Running necery test. Case
(github Actions server)
Ci.yaml
Jenkins
Unit--test--Cases
Continues integration
ggt
DEV
UNTI TEST CASES
INEGRATION TEST CASES
Dev
Testing Phase
Specific team
TEST CASES

UNIT TEST CASES

1. TEST

→ unit_test.py ← this file for writing a test cases

1. Before Pushing the changes to github or bitbucket
2. After Pushing the changes to github or bitbucket



Unit test cases will be available

ci.yaml

{ test case execution would be on the Github Action Server }

Assignments

Note: Setup is done for test cases (unit test case)

1. You have to write at least 10 unit test cases

You have to do — mandatory for all

test folder → (unit_test.py)

Optional Assignment → Unit test ✓

Integration test cases ✗ You have to do

this is done

CI → After pushing the code over the github we are running the test case ✓

Next is → Before pushing the code over the github run the test cases ✗

You have to do

Pre commit test cases

Now Aws Deploy

1. ECR
2. IAM USER
3. ECS
4. Aws SECRATE

Services over the Aws

(this is not recommended Practice)

1. Root user
2. IAM USER

Accen
- Polices
- Roles

Credentials

github Action → Aws

1. Aws Access key ID
2. Aws Accen Secrate key

Security Credentials

certification → security credential

Root user → Aws Account ← GOD

Root user credential → Complete Accen of Aws ← GOD

Aws

server

EC2

ECS

- AGENT ← AI

Security credentials

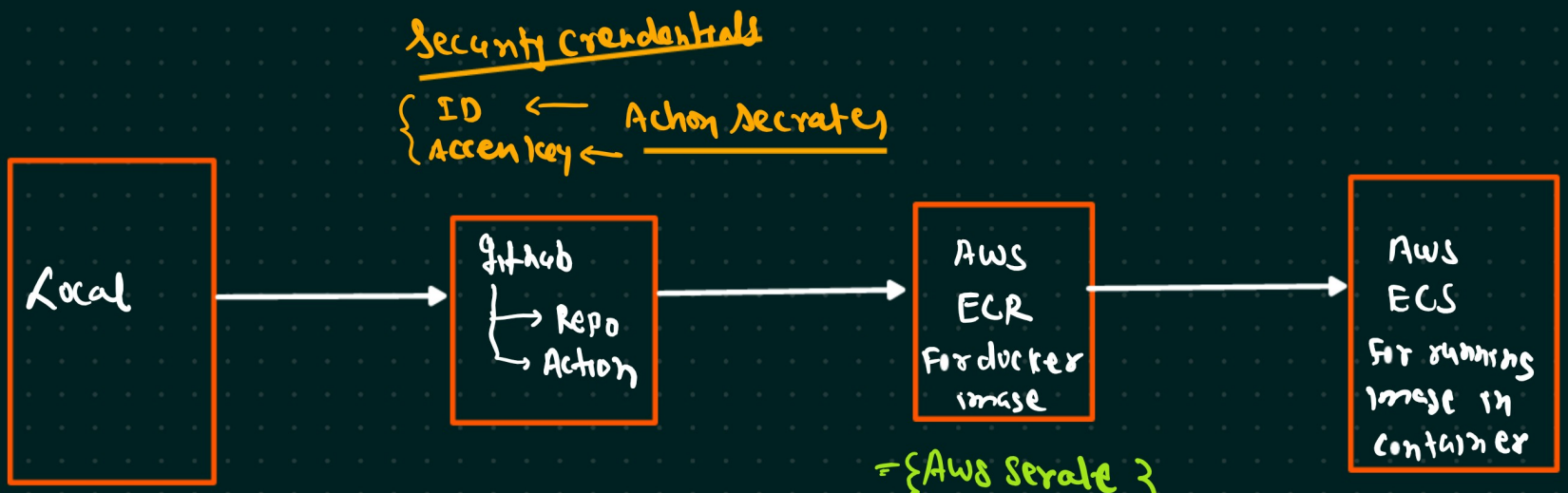
Root - SC X

= IAM_USER - given Specific permission to that user

↓

Security credentials from here

1. ECR Repo
2. IAM USER



- Aws Serate manager
- API keys

- ci.yaml ← testing
- aws.yaml ← CD/CD
- task-definition ← ECS
- .env ← API keys

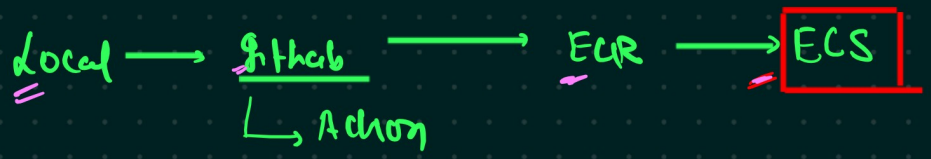
logs → cloudwatch
ECS logs

Later on ← S3 ← (logs)
← (Artifacts ← Data)
session/faiss index

aws.yaml
task-definition.json

- Cloudformation

optional ← DynamoDB
{key: value
}



1. Code level changes → {task definition, model-loder.py}
   → aws.sam2
   - ci.yml test
   → aws.yaml
   → task_definition.json

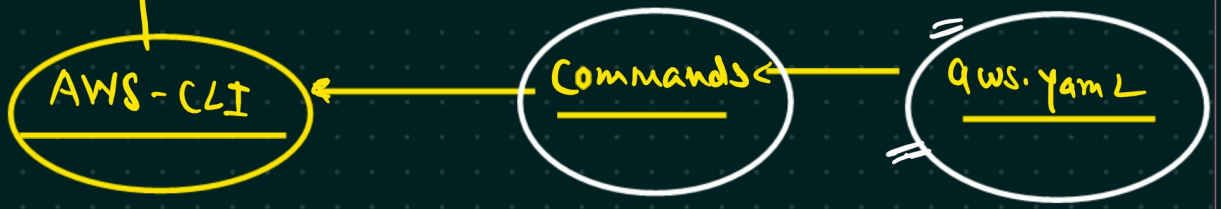
ECS
=

CLI ← Command line interface

AWS CLI ← You can install aws CLI AND Execute the Command on to it-

github ← org

- Repo
- Action ← server ← (ubuntu-latest)



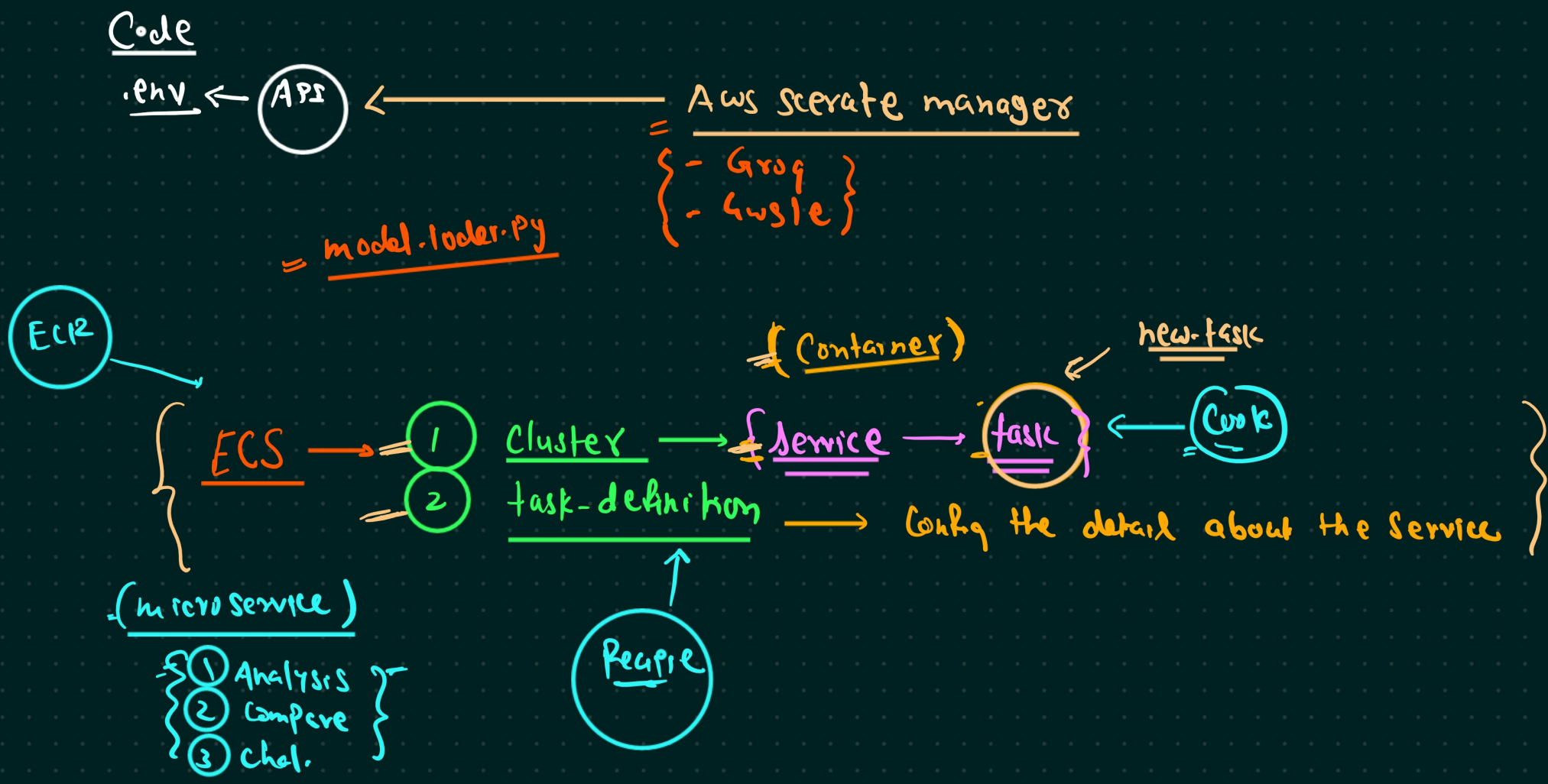
X we are not using

Self-hosted-Runner

github

- Action ← server ← Your own server here

Local system
third party server
Custom EC2 ← AWS


Code
.env
API
Aws scerate manager
- Grog
- Awsle
= model.loder.py
ECR
ECS
1
2
Clustey
task-definition
(Container)
Service
task
new-task
Cook
Config the detail about the Service
(micro service)
1 Analysis
2 Compere
3 Chal.
Recipe

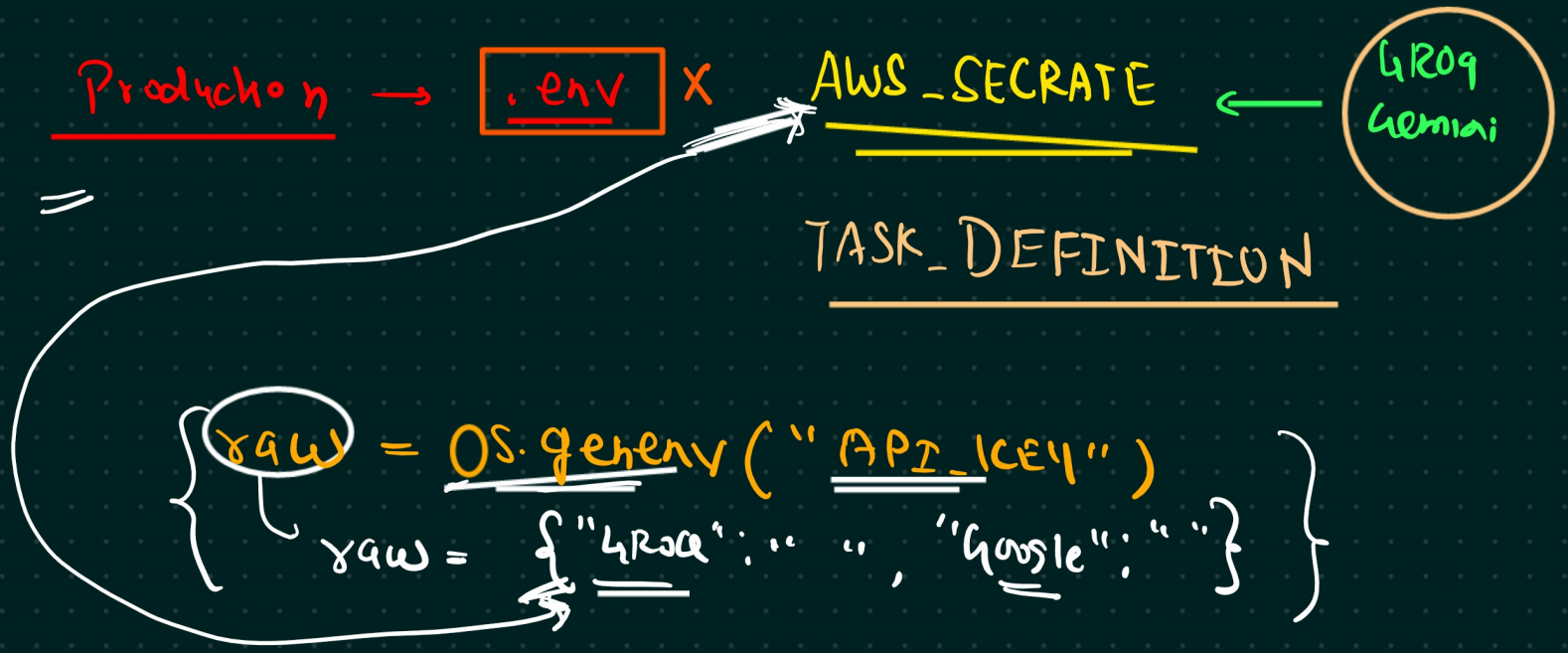
MODEL → GROQ, GEMINI

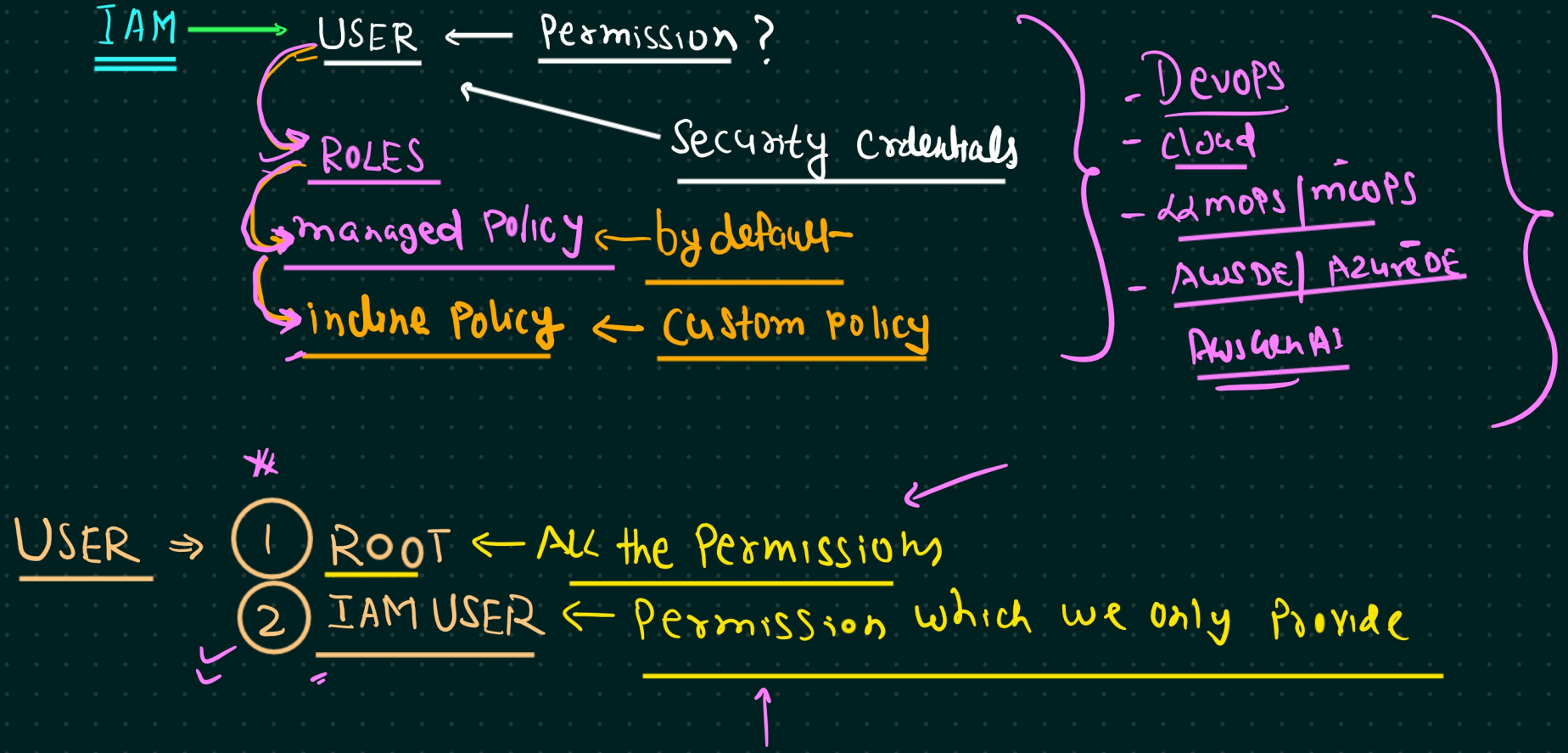
API KEY

Local ← .env ← ?

API-KEY

key = os.getenv(" ")




IAM
USER
Permission ?
Security Credentials
ROLES
managed Policy
by default
inline Policy
Custom policy
- DevOps
- Cloud
- MLOPS | MCOPS
- AWS DE | AZURE DE
AWS GEN AI
USER
1
ROOT
All the Permissions
2
IAM USER
Permission which we only Provide

ECR =

AWS SECRET MANAGER :-

ECS :-