I don't have a test environment for Azure, so all actions related to Azure (except Azure Devops) are theoretical in nature without practical testing.

1. Part 1. CI/CD Promotion Flow • GitHub Actions + Azure Pipelines

   For the test task, I used the following build/deployment scheme: when merging to the main branch, a Docker image of the application is created and pushed to the ECR (by Github Action, authorization via OIDC GITHUB). On the ECR side, a vulnerability check occurs. Azure Devops is also initiated, where this image can be deployed to either production (AWS) or dev (Azure). During deployment, the newly launched image is health-checked. If the health check is successful, the deployment is completed successfully. If the health check is unsuccessful, the last working version is rolled back and the deployment is completed slowly.

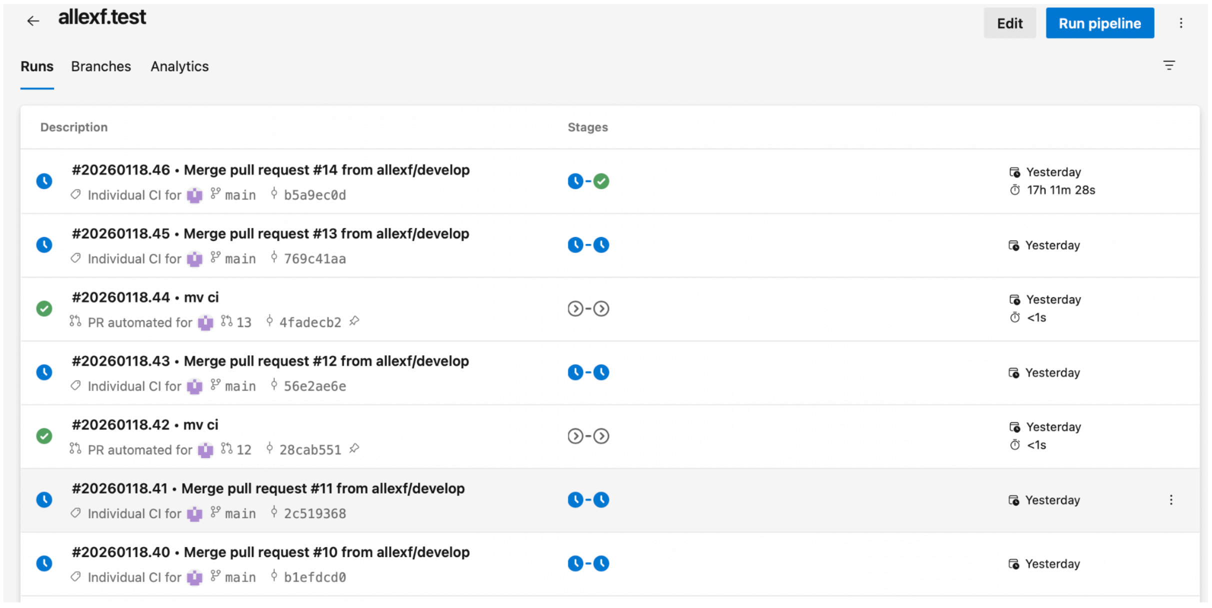
   In real life, I usually adhere to the following scheme: there are dev and main branches. When committing to the dev branch, an automatic build and deployment to the dev environment occurs; when merging from the dev branch to the main branch, an automatic build and deployment to the staging environment occurs. When a tag is created from the main branch, the image with the tag is built and a step is created to confirm the deployment to production.

   Accessing the application through the balancer:

   ```
   curl -ik test-app-prod-2106904593.eu-north-1.elb.amazonaws.com/healthz
   HTTP/1.1 200 OK
   Date: Mon, 19 Jan 2026 08:35:16 GMT
   Content-Type: text/plain; charset=utf-8
   Content-Length: 2
   Connection: keep-alive
   server: uvicorn

   OK
   ```

   Azure Devops pipeline screen:


allexf.test

Description	Stages	
#20260118.46 • Merge pull request #14 from allexf/develop Individual CI for main b5a9ec0d		Yesterday 17h 11m 28s
#20260118.45 • Merge pull request #13 from allexf/develop Individual CI for main 769c41aa		Yesterday
#20260118.44 • mv ci PR automated for 13 4fadecb2		Yesterday <1s
#20260118.43 • Merge pull request #12 from allexf/develop Individual CI for main 56e2ae6e		Yesterday
#20260118.42 • mv ci PR automated for 12 28cab551		Yesterday <1s
#20260118.41 • Merge pull request #11 from allexf/develop Individual CI for main 2c519368		Yesterday
#20260118.40 • Merge pull request #10 from allexf/develop Individual CI for main b1efdcd0		Yesterday

2. Part 2. Infrastructure as Code • Terraform

   I didn't use Terraform in the deployment. In my scenario, Terraform runs once at the very beginning. The state is stored in a pre-created S3 bucket (the lock can also be stored in dynamodb). The file terraform.tfstate is attached to infra/envs/prod-aws. I used the following commands to start Terraform:

```
cd infra/envs/prod-aws

terraform init \
  -backend-config="region=eu-north-1" \
  -backend-config="profile=YYYY" \
  -backend-config="bucket=ZZZZZZ"

terraform apply \
  -var="aws_account_id=XXXXXXX" \
  -var="aws_region=eu-north-1" \
  -var="aws_profile=YYYY" \
  -var="github_login=allexf" \
  -var="github_repo=test" \
  -var="vpc_id=vpc-XXXXX" \
  -var='subnet_ids=["subnet-XXXXX","subnet-YYYYY"]' \
  -var="rds_enabled=false" \
  -var="project=test-app" \
  -var="environment=prod" \
  -var="ec2_subnet_id=subnet-ZZZZZZ" \
  -var="ami_id=ami-08e5dcf7e45dea85e" \
  -var="ssh_key_name=test-16-01-26" \
-var="instance_type=t4g.small" \
-var="static_site_bucket_name=myapp-test-egee1oor5"
```