CDW Documentation

Azure Github Actions Pipeline

Azure Github Actions Pipeline

Demo Objective

Create an automated CI/CD pipeline using **GitHub Actions** that:

- Trains and registers a machine learning model
- Deploys it to a managed Azure endpoint
- Enables full **observability**: logging, alerts, and diagnostics

---

Key Infrastructure Components

Provision these with **Bicep** or **Terraform**:

1. **Azure ML Workspace**
2. **Azure Key Vault** (for secrets like storage keys)
3. **Azure Storage Account** (for data input/output)
4. **Azure Container Registry** (optional: custom container inference)
5. **Azure Application Insights** (for logs and metrics)
6. **Azure Monitor Alert Rules** (trigger on failed jobs or degraded endpoints)
7. **Compute cluster** (for training, e.g., `cpu-cluster`)
8. **Azure ML Online Endpoint** (for model deployment)

Repo Structure

```
plaintextCopyEdit.github/workflows/
├── train-deploy.yml         # GitHub Actions workflow
infra/
├── main.bicep               # Infrastructure as code
ml/
├── train.py                 # Model training script
├── score.py                 # Inference entry point
├── environment.yml          # Conda environment for training/deployment
├── register_model.py        # Registers trained model
├── pipeline_job.yml         # Azure ML pipeline definition (optional)
```

CI/CD Flow (via GitHub Actions)

Trigger: Push to main or model-update branch

1. **Checkout & Install Dependencies**
2. **Login to Azure** (`azure/login` GitHub Action)
3. **Set up Azure ML CLI or Python SDK**
4. **Run Training Script** (optionally via pipeline YAML)
5. **Register Model** to Azure ML Registry
6. **Deploy Model** to Online Endpoint
7. **Post-deployment Tests**
8. **Publish Logs** to Application Insights
9. **Trigger Alerts** if any step fails (via `az monitor` alert rules)

Logging, Monitoring & Alerts

- **Application Insights**: attach to the Azure ML endpoint for request/response logs and metrics.
- **Azure Monitor Alerts**:
  - Alert on failed training runs (via log analytics query).
  - Alert on high latency or low success rate on the deployed endpoint.
  - Notification via email/webhook/Teams.

Demo Enhancements

1. **Dashboards**: Include an Azure Dashboard that surfaces training job status, endpoint performance, recent alerts.
2. **Web Frontend** (Optional): Simple app to send inference requests, visualize logs.
3. **Cost Control**: Auto-shutdown training compute after use.

Example Scenario

Business Case: Retrain a churn prediction model every week using new customer data.

- GitHub Actions scheduled trigger: weekly
- Logs retraining results
- Deploys model if metrics (e.g., accuracy > previous version) pass
- Sends alerts if model accuracy drops >10% or job fails

Security Considerations

- Use GitHub Secrets for Azure credentials

- Leverage **Workload Identity Federation** for GitHub-Azure auth
- RBAC for least-privilege access to ML and monitoring resources

---

Success Criteria

- CI/CD pipeline runs end-to-end on commit
- Azure infrastructure deployed from code
- Model available at a public or private endpoint
- Logs visible in App Insights
- Alerts trigger on defined failure conditions