

PORTFOLIO DEVOPS –CLOUD AZURE

2025

Projet : REPOSITORY MASTER TEMPLATE IAC/CD

Projet : WORKFLOW TERRAFORM

Projet : MONITORING TERRAFORM

Projet : ZERO TRUST

Projet : GESTION BACKEND

Projet : PIPELINE CI/CD DOCKER MODULAIRE ET SÉCURISÉE

Projet : DÉPLOYER EN CONTINU UNE APPLICATION

Projet : DÉPLOIEMENT OUTILS VMS

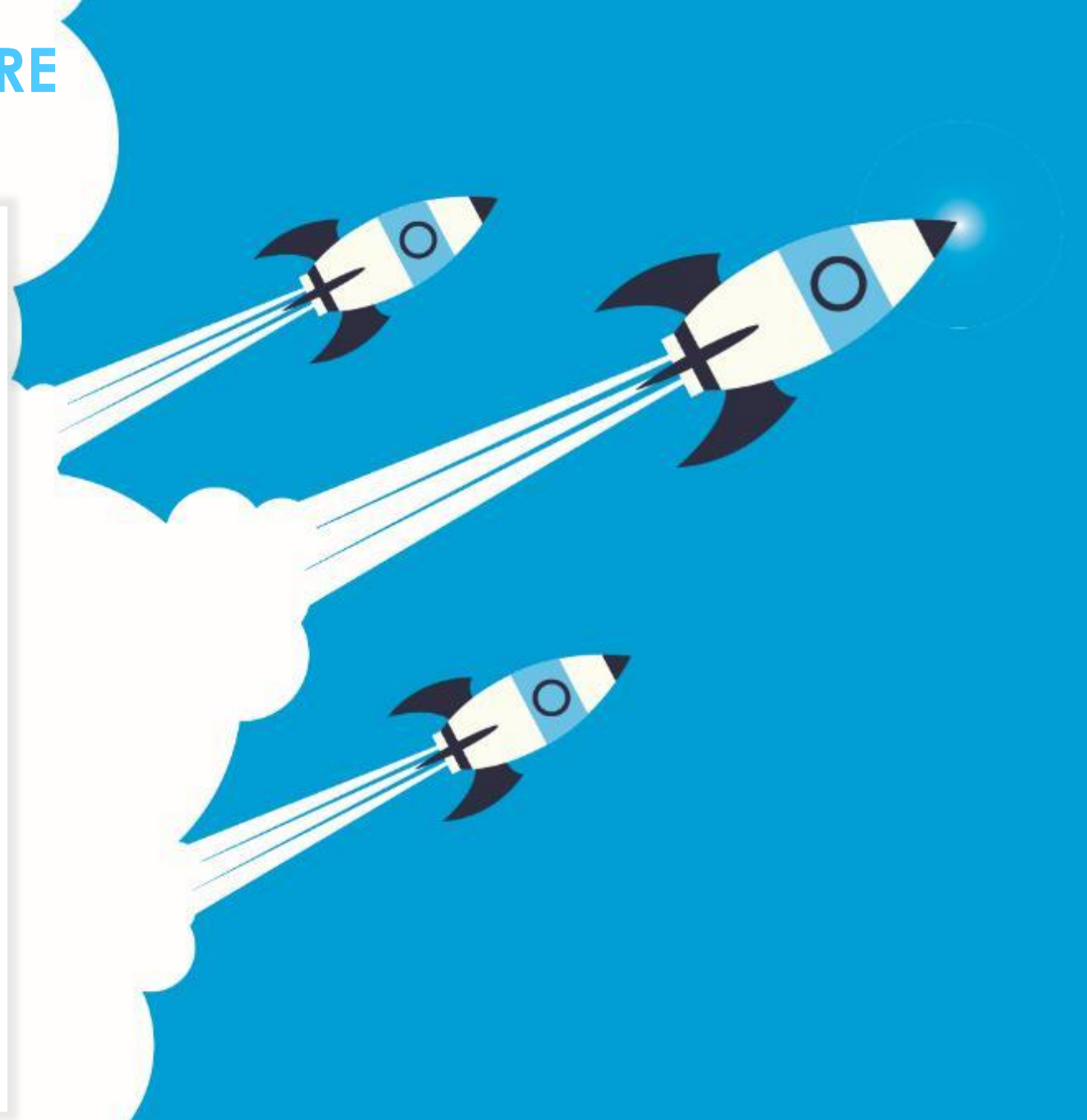
Projet : AUTOMATISATION SERVICE CONNECTIONS AZURE

Projet : DUMP DATA BASE POSTGRE

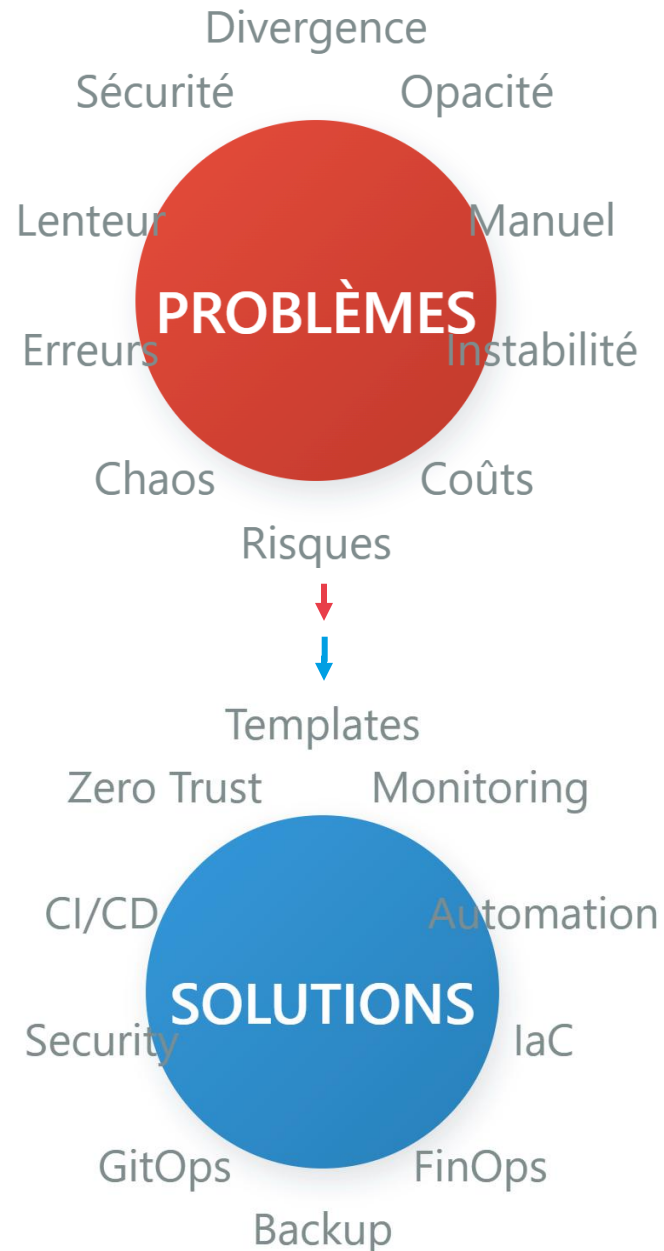
Projet : TEMPLATE DÉPLOIEMENT K8S GÉNÉRIQUE

Projet : AUTOMATISATION SCAN SÉCURITÉ TRIVY K8S

Projet : AUTOMATISATION INFRASTRUCTURE AKS



CHAQUE PROJET RÉPOND À UN BESOIN CONCRET CLIENT/INTERNE



Standardisation & Cohérence

Repository Master Template → Éviter la divergence entre projets

Workflow Terraform → Uniformiser les déploiements multi-équipes

Visibilité & Observabilité

Monitoring Terraform → Anticiper les pannes infrastructure

Dump Database Postgre → Sauvegardes automatisées fiables

Sécurité & Conformité

Zero Trust → Éliminer les accès non contrôlés

Scan Sécurité Trivy K8S → Détecter vulnérabilités en amont

Accélération Delivery

Pipeline CI/CD Docker → Réduire time-to-market applications

Déploiement Continu → Livraisons fréquentes sans risque

Template K8S Générique → Déploiements reproductibles

Réduction Tâches Manuelles

Automatisation Service Connections → Fin de la config manuelle

Déploiement VMs → Provisioning en un clic

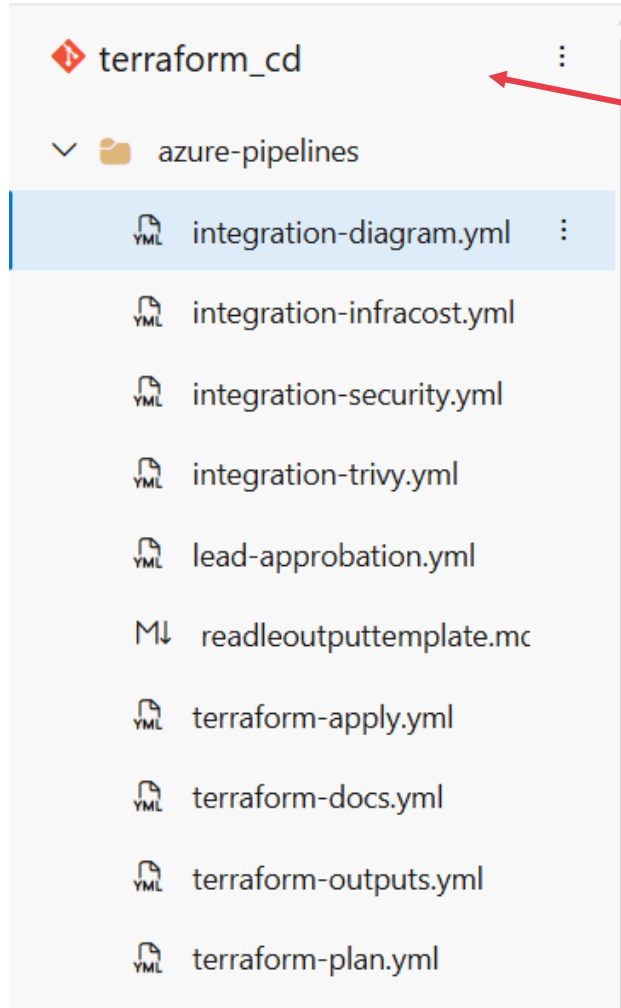
Infrastructure AKS → Clusters Kubernetes standardisés

Gestion d'État & Collaboration

Gestion Backend → Travail équipe sécurisé sur infrastructure

REPOSITORY MASTER TEMPLATE

❖ MASTER REPOSITORY



Structure Repositories :

2 repositories :

`terraform_cd` (pipelines)

`terraform_iac` (modules)

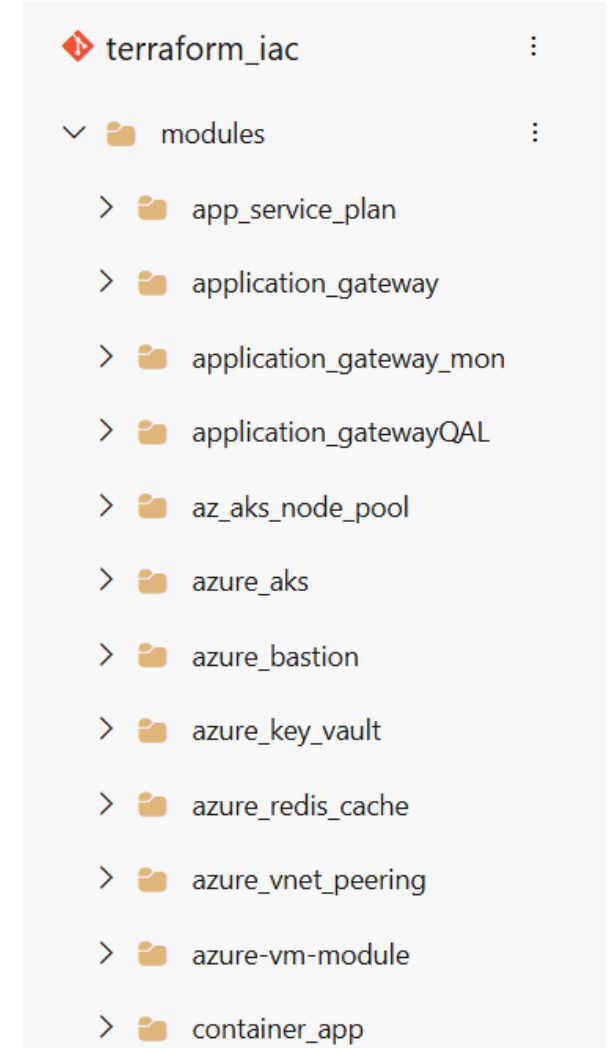
Architecture d'entreprise:

Templates modulaires : 1 template = N projets

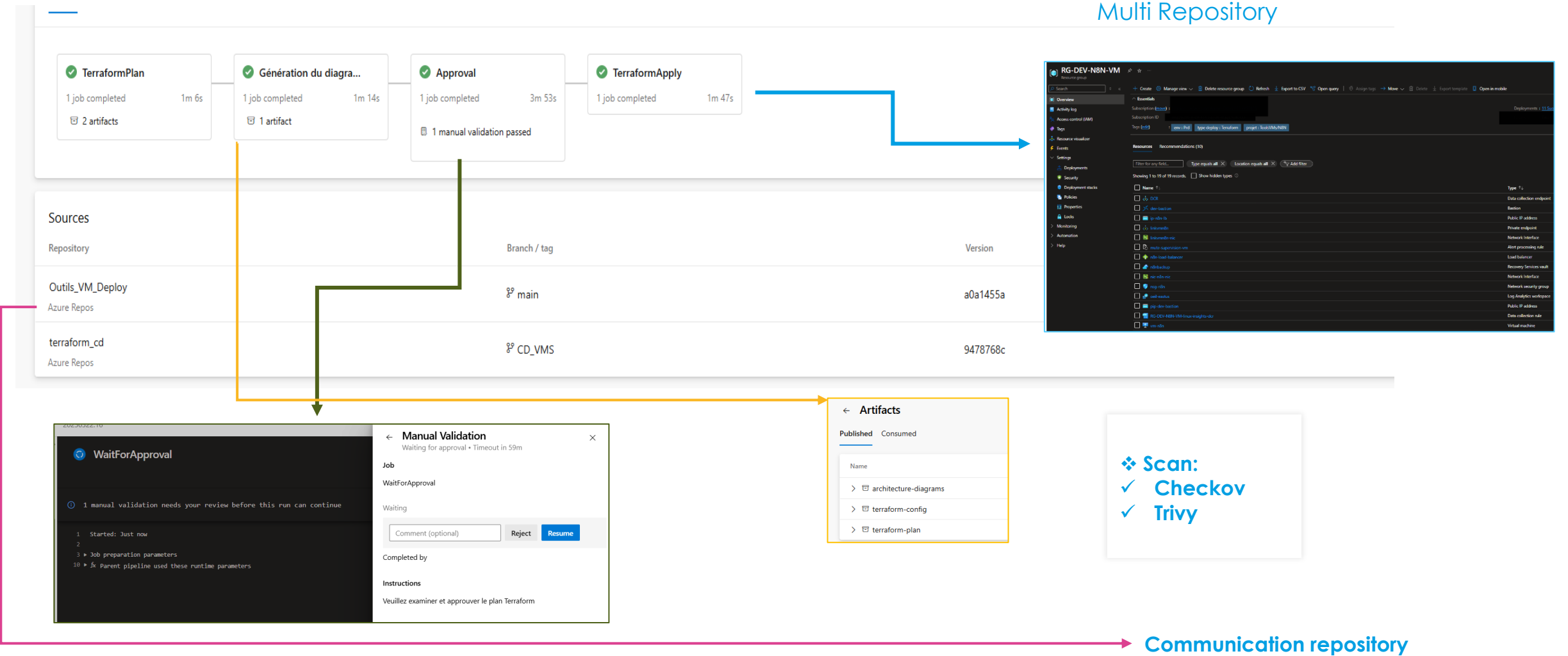
Paramètres dynamiques : Multi-environnements

Convention nommage : Standardisée et cohérente

Intégrations GitOps : Pipeline as Code complet

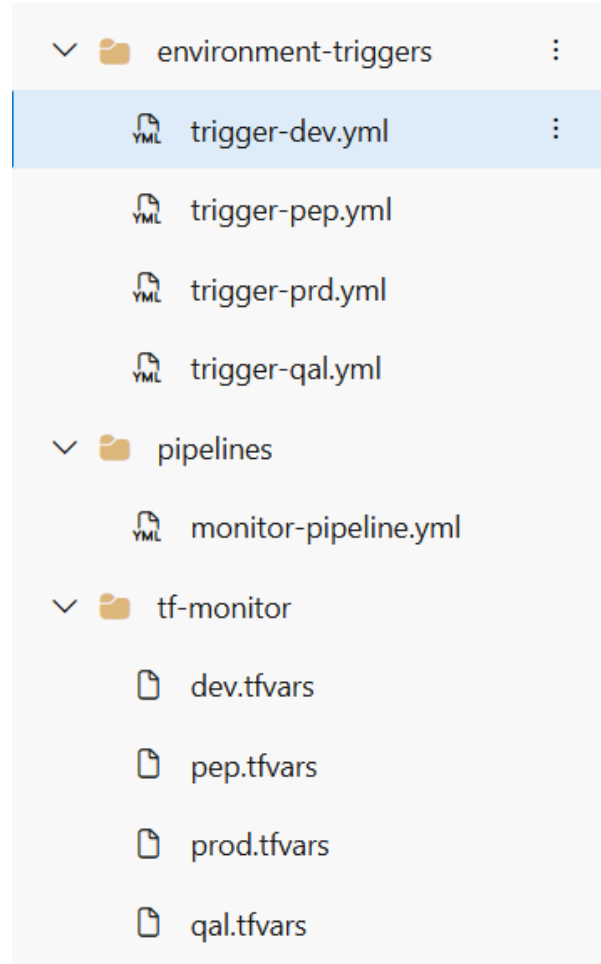


WORKFLOW TERRAFORM



MONITORING TERRAFORM

❖ Pipeline Terraform modulaire multi client et sécurisée



Use Cases Supportés :

- ✓ Analyses conditionnelles : Sécurité + Coûts + Diagrammes optionnels
- ✓ Multi-environnements : DEV/PRD avec triggers dédiés
- ✓ Monitoring AppGW : Métriques + alertes module X
- ✓ Architecture as Code : Génération diagrammes automatique
- ✓ FinOps intégré : Analyse coûts Infracost dans la pipeline
- ✓ Templates modulaires : Extension de pipelines partagées

KPI Mesurés :

- ✓ Paramètres configurables : Contrôle fin des analyses
- ✓ Stages conditionnels : Plan → Security → Cost → Diagram → Apply
- ✓ Intégrations avancées : Security, Infracost, Architecture
- ✓ Environment triggers : Automatisation par environnement
- ✓ Module monitoring : AppGW métriques + Action Groups
- ✓ Convention nommage : \${Company}\${Language}-\${App}-\${Environment}

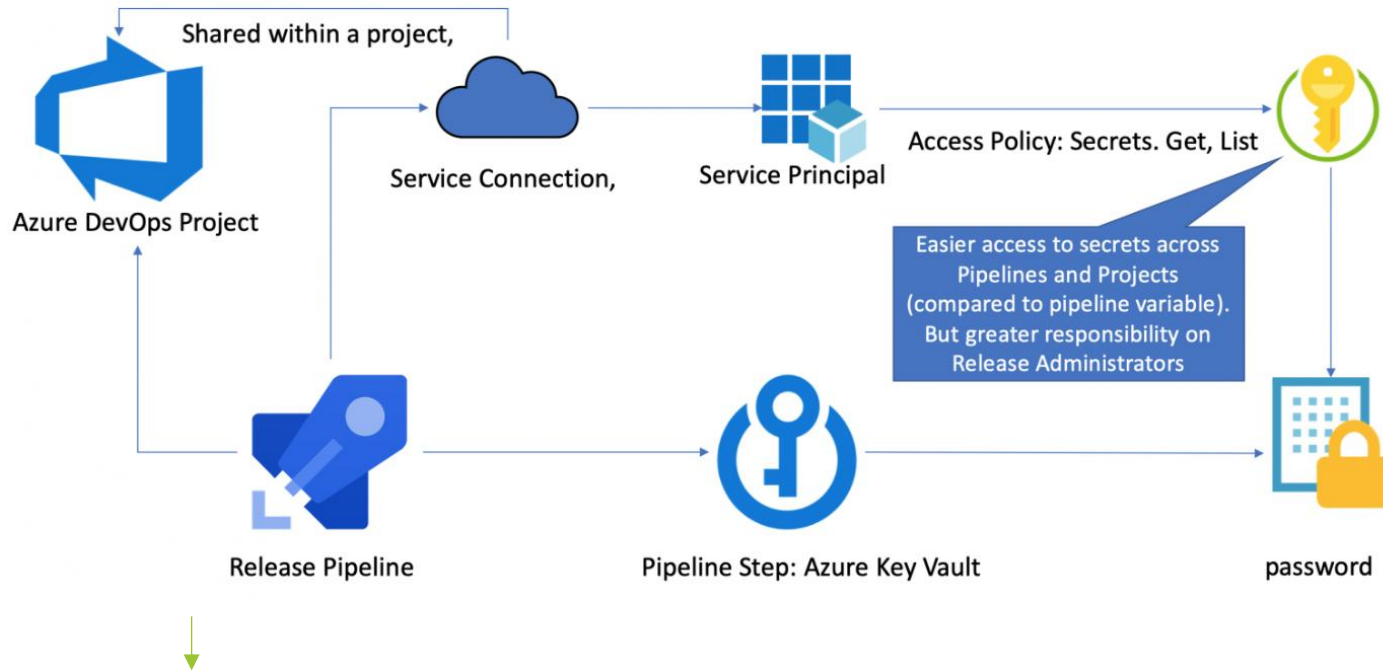
Appel module Dynamiques

- Log_analytics
- Action group
- Alert rules

```
module "appgw_monitoring" {
  source = "git::https://@dev"
  resource_group_name = "rg-${var.Company}${var.Language}-${var.ApplicationName}-${var.Domain}"
  action_group_rg      = "rg-${var.Company}${var.Language}-${var.ConfigurationItem}"
  action_group_name    = "ag-${var.Company}${var.Language}-${var.ApplicationName}"
  application_gateway_name = "agw-${var.Company}${var.Language}-${var.ApplicationName}-${var.Environment}"
  application_gateway_metric_alerts = var.application_gateway_metric_alerts
  application_gateway_query_alerts = var.application_gateway_query_alerts
}
```

ZERO TRUST

❖ KEY VAULT/AZURE DEVOPS/SERVICE CONNECTION




Étapes de pipeline utilisant Key Vault :

- ❖ Pré-tâche Get SSL Cert : récupération certificats
- ❖ Téléchargement automatique secrets : injection variables Déploiement
- ❖ N8N stack : utilisation secrets sans exposition

- ❖ Récupération des secrets dans le keyvault
- ❖ zack.identity avec permissions "Get, List" sur falloutkvault
- ❖ Agent VM azure DevOps
- ❖ Service Principal configuré

```
Key vault name: falloutvault. ⇄  
Downloading secret value for: prod-n8n-password.  
Downloading secret value for: prod-postgres-password.
```

```
Key vault name: falloutvault.  
Downloading secret value for: n8n-ssl-cert. ⇄  
Downloading secret value for: n8n-ssl-key.  
Finishing: 🗝️ Get SSL Certificates from falloutvault
```

 **azurekeyvaultsecretsprovider**
Managed Identity

```
pool:  
  name: 'vm-n8n-agents'  
jobs:  
  - deployment: DeployJob  
    environment: '${targetEnvironment}'  
  -  
    azureSubscription: 'az_connection'
```

GESTION BACKEND

❖ SÉCURISER L'INFRASTRUCTURE

GESTION BACKEND

SÉCURITÉ DU STATE

- Verrouillage automatique (lock)
- Accès contrôlé par Service Principal
- Pas de secrets en clair dans le state

> List Storage Account Keys	Succeeded	6 minutes a...	Fri May 23 2...	[REDACTED]	zak-sp
> List Storage Account Keys	Succeeded	6 minutes a...	Fri May 23 2...	[REDACTED]	zak-sp
> List Storage Account Keys	Succeeded	7 minutes a...	Fri May 23 2...	[REDACTED]	zak-sp

<input type="checkbox"/>	tfstate.n8nenvdev
<input type="checkbox"/>	tfstate.vmn8ndev
<input type="checkbox"/>	tfstate.vmn8ndevenvdev
<input type="checkbox"/>	tfstate.vmazureprod
<input type="checkbox"/>	tfstate.vmazureprodenvdev
<input type="checkbox"/>	tfstate.vmazureprodenvprd
<input type="checkbox"/>	tfstate.webapps
<input type="checkbox"/>	tfstate.webappsenvdev

- WORKSPACE ISOLATION
tfstate séparé par environnement
- - dev : tfstate.vpdev
- - prd : tfstate.vmprd

State partagé
collaboration équipe

Terraform

❖ State

```
terraform {  
  required_providers {  
    azurerm = {  
      source = "hashicorp/azurerm"  
      version = "3.44.1"  
    }  
  }  
  backend "azurerm" {  
  }  
}  
  
provider "azurerm" {  
  features {  
  }  
}
```

- Provider.tf

- Storage Account
- Soft delete 7

Blob service	
Hierarchical namespace	Disabled
Default access tier	Hot
Blob anonymous access	Disabled
Blob soft delete	Enabled (7 days)
Container soft delete	Enabled (7 days)
Versioning	Disabled
Change feed	Disabled
NFS v3	Disabled
Allow cross-tenant replication	Disabled
Storage tasks assignments	None

AUTOMATISATION SERVICE CONNECTIONS AZURE

❖ Résultat Pipeline modulaire réutilisable et refactoring facilité

Use Cases Supportés :

- ✓ Multi-clients : Isolation complète par tenant et sub ID
- ✓ Multi-environnements : DEV/STAGING/PROD
- ✓ Renouvellement auto : Certificats + secrets
- ✓ Validation temps réel : Format GUID + permissions
- ✓ Gestion des erreurs

KPI Mesurés :

- ✓ 1min 21sec : Temps création Service Connection
- ✓ 95% réduction vs processus manuel (30min)
- ✓ 100% secrets rotés automatiquement
- ✓ Plusieurs clients utilisateurs actifs

Paramètres modulaires - Fonctionnalités optionnelles

- name: EnableSecretCheck
 displayName: 'Vérifier expiration des secrets'
 type: boolean
 default: false
- name: EnableSecretRenewal
 displayName: 'Autoriser renouvellement des secrets (nécessite vérification)'
 type: boolean
 default: false
- name: EnableServiceConnectionCreation
 displayName: 'Créer Service Connection'
 type: boolean
 default: true

Référence aux repos externes

resources:

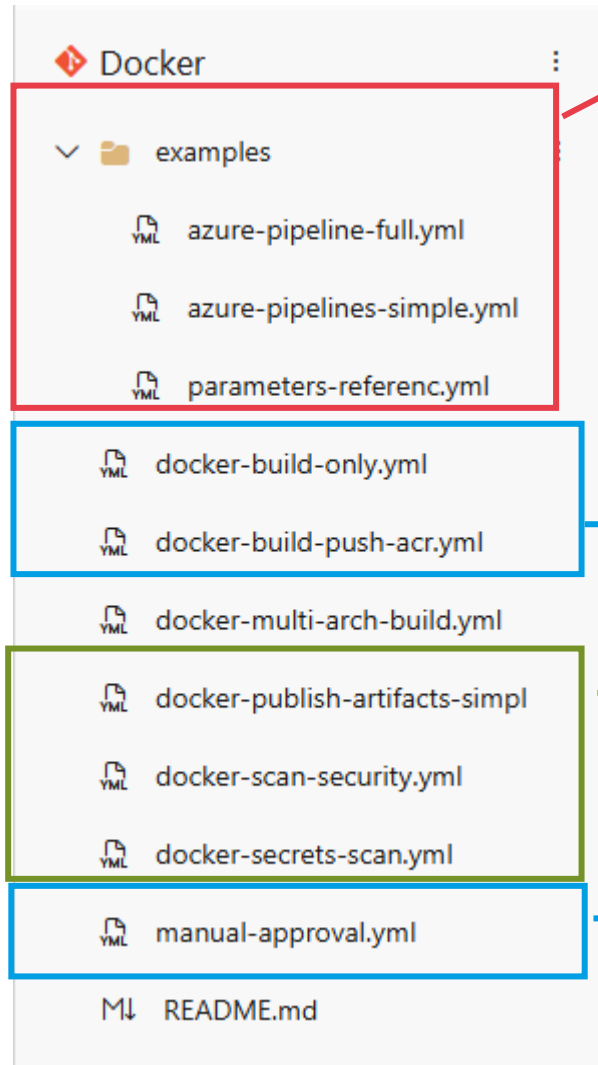
repositories:

- repository: master_repo
 type: git
 name: TEST-ServiceCO
- repository: azure_cli_scripts
 type: git
 name: TEST-ServiceCO-Script

```
79  ✓ Service Connection 'AppReg [redacted] ' créée avec succès
80
81  Finishing: Create Service Connection
```


PIPELINE GÉNÉRIQUE DOCKER

❖ PIPELINE CI/CD DOCKER MODULAIRE ET SÉCURISÉE



Use Cases Supportés :

- ✓ Build modulaire : Build-only local + Build+Push ACR séparés
- ✓ Sécurité Docker : Scan Trivy images avec dashboard Junit
- ✓ Tests automatisés : Validation startup + sécurité + layers
- ✓ Templates génériques : Réutilisables pour toutes applications
- ✓ Approbation manuelle : Contrôle déploiement environnements
- ✓ Publication artifacts : Build info + security reports

KPI Mesurés :

- ✓ templates Docker : Build-only, Build+Push, Security scan
- ✓ paramètres configurables : Dockerfile, repository, context, args
- ✓ Tests de base : Startup + user check + functionality
- ✓ Scan vulnérabilités : CRITICAL/HIGH/MEDIUM/LOW comptées
- ✓ Reports JUnit : Dashboard intégré Azure DevOps Tests
- ✓ Approbation manuelle : Pipeline sécurisée production

DÉPLOYER EN CONTINU UNE APPLICATION

❖ Variables par branche

```
55 # Variables dynamiques par branche
56 ${ if eq(variables['Build.SourceBranch'], 'refs/heads/N8N.PRD') }}:
57   targetEnvironment: 'n8n-production'
58   deploymentName: 'Production'
59 ${ if eq(variables['Build.SourceBranch'], 'refs/heads/N8N.DEV') }}:
60   targetEnvironment: 'n8n-development'
61   deploymentName: 'Development'
62
```

❖ vm-n8n-agents pour isolation et environnement

```
dependsOn: Approval
pool:
  name: 'vm-n8n-agents'
jobs:
  - deployment: DeployJob
    environment: '${targetEnvironment}'
```

❖ Secrets sécurisés

```
# === ÉTAPE 1 : Récupération SSL (toujours) ===
- task: AzureKeyVault@2
  displayName: '🔑 Get SSL Certificates from falloutvault'
  inputs:
    azureSubscription: 'az_connection'
    KeyVaultName: 'falloutvault'
    SecretsFilter: 'n8n-ssl-cert,n8n-ssl-key'
    RunAsPreJob: false
```

❖ Connexion docker

```
# === ÉTAPE 3 : Login Docker ACR ===
- task: Docker@2
  displayName: '🌐 Login to ACR'
  inputs:
    containerRegistry: '${dockerRegistryServiceConnection}'
    command: 'login'
```

❖ Isolation variables environnement

```
# === ISOLATION TOTALE PAR PROJET DOCKER COMPOSE ===
echo "🔒 Setting up complete Docker isolation..."
export COMPOSE_PROJECT_NAME="n8n-${CLIENT_ID}"

echo "📋 Isolation Settings:"
echo "  COMPOSE_PROJECT_NAME: $COMPOSE_PROJECT_NAME"
echo "  CLIENT_ID: ${CLIENT_ID}"

# === Pull latest images ===
echo "📦 Pulling latest images..."
docker-compose -p "$COMPOSE_PROJECT_NAME" -f docker-compose.generated.yml pull

# === Start services AVEC ISOLATION ===
echo "🚀 Starting N8N stack with SSL and PERFECT ISOLATION..."

# ✅ CLEF DU SUCCÈS: Projet dédié
docker-compose -p "$COMPOSE_PROJECT_NAME" -f docker-compose.generated.yml up -d postgres n8n nginx
```

- ❖ Variables dynamiques : détection automatique branche → environnement
- ❖ Déploiement safe : arrêt sélectif + vérification de préservation
- ❖ Logs et health checks par environnement
- ❖ Vm-n8n-agents pour isolation

DÉPLOIEMENT OUTILS VMS

Run pipeline

Select parameters below and manually run the pipeline

Branch/tag

main

Select the branch, commit, or tag

Environnement cible

☒ dev

☐ prd

Type de système d'exploitation

☒ linux

☐ windows

☒ Déployer la VM

☒ Déployer le Bastion

☐ Déployer une IP publique

☒ agent AMA

☐ AUM avec policy

☐ Activer la sauvegarde

☐ Exécuter l'analyse de sécurité

☐ Exécuter l'analyse des coûts

☒ Générer diagramme d'architecture

Advanced options

Environnement

OS

Outils_VM_Deploy

tfvars

dev.tfvars

linux.tfvars

prd.tfvars

windows.tfvars

```
Création ou sélection du workspace dev...  
Switched to workspace "dev".  
Génération du plan Terraform avec les arguments: -var=deploy_vm=true -var=deploy_bastion
```

```
rerun this command to reinitialize your working directory. If you forget, other  
commands will detect it and remind you to do so if necessary.  
Switched to workspace "prd".
```

Choix des paramètres

```
160 module.bastion[0].azurerm_bastion_host.bastion: Still creating... [09m40s elapsed]  
161 module.bastion[0].azurerm_bastion_host.bastion: Creation complete after 9m44s  
162  
163 Apply complete! Resources: 4 added, 0 changed, 0 destroyed.  
164  
165 Outputs:  
166  
167 bastion_host_id = "/subscriptions/***/resourceGroups/RG-DEV-N8N-VM/providers/  
168 debug_vars = {  
169   "deploy_bastion" = true 1  
170   "deploy_public_ip" = false 2  
171   "deploy_vm" = true 3  
172   "enable_backup" = false 4  
173   "enable_monitoring" = false 5  
174   "enable_update_management" = false 6  
175   "use_modern_agents" = true 7  
176   "vm_os_type" = "linux" 8  
177 }  
178 vm_ids = tolist([  
179   "/subscriptions/***/resourceGroups/RG-DEV-N8N-VM/providers/Microsoft.Compute  
180 ])  
181 vm_names = tolist([  
182   "vm-n8n",  
183 ])  
184 vm_private_ips = tolist([  
185   "10.0.0.4",  
186 ])  
187 vm_public_ips = []  
188  
189 Finishing: Terraform Apply
```

DUMP DATA BASE POSTGRE

❖ DUMP=> STORAGE ACCOUNT

↓

Stages Jobs

✓ Backup N8N Data...

1 job completed 58s

1 artifact

✓ Validate Backup

1 job completed 25s

1 artifact

✓ Cleanup Old Back...

1 job completed 24s

❖ STORAGE ACCOUNT => CONTAINER => DUMP

↓

\$logs

database-backups

dump

keyrotationlog

saslog

View all

File shares

Queues

Tables

Showing all 8 items

Name	Last modified
[.]	
backup_metadata_1706.json	01/06/2025
backup_metadata_1713.json	01/06/2025
backup_metadata_1727.json	01/06/2025
backup_metadata_1732.json	02/06/2025
n8n_prod-client_backup_202506...	01/06/2025
n8n_prod-client_backup_202506...	01/06/2025
n8n_prod-client_backup_202506...	01/06/2025
n8n_prod-client_backup_202506...	02/06/2025

❖ PIPELINE RESTORE DATA BASE

❖ PARAMETERS RESTORE

Run pipeline

Select parameters below and manually run the pipeline

Branch/tag

N8N.PRD

Select the branch, commit, or tag

Environnement cible pour la restauration

development

production

Date du backup (YYYYMMDD_HHMMSS ou "latest" pour le plus récent)

20250602_092447

Environnement source du backup

development

production

☒ ⚠ JE CONFIRME la restauration (TOUTES les données actuelles seront PERDUES)

Raison de la restauration

'Restauration depuis backup PROD - rollback incident'

Advanced options

Variables

This pipeline has no defined variables

Stages to run

Run as configured

Resources

Use latest version of all resources

☐ Enable system diagnostics

Cancel

Run

⚠ PRODUCTION RESTORE APPROVAL REQUIRED

Restore Details:

Target: PRODUCTION (prod-client)

Source: production (prod-client)

Backup: 20250602_092447

Reason: 'Restauration depuis backup PROD - rollback incident'

⚠ This will DESTROY current production data!

✓ Approved by: Zack BOUREDJI

Checks and manual validations for Manual Approval for Production

Permission Environment n8n-production-restore

Permission needed

Permit

```

1 parameters:
2 - name: BackendServiceConnection
3   type: string
4 - name: kubectlVersion
5   type: string
6   default: '1.32.3' #latest
7 - name: kubeloginVersion
8   type: string
9   default: 'v0.2.7' #latest
10 - name: aks_name
11   type: string
12 - name: aks_rg
13   type: string
14 - name: k8sNamespace
15   type: string
16   default: 'kube-system'
17 - name: resource
18   type: string
19   default: 'AzureIngressProhibitedTargets'
20 - name: manifest
21   type: object
22   default: manifest.yaml

```

```

! - task: KubeloginInstaller@0
  displayName: 'install kubelogin'
  inputs:
    kubeloginVersion: ${ parameters.kubeloginVersion }

- task: Kubernetes@1
  displayName: Login to Kubernetes cluster ${ parameters.aks_name }
  inputs:
    connectionType: 'Azure Resource Manager'

```

```

kubectl get ${ parameters.resource } -n ${ parameters.k8sNamespace } -o json \
| jq -r ' [.items[] | {name:.metadata.name, hostname:.spec.hostname, paths:.spec.paths, creationTimestamp:.metadata.creationTimestamp}] '

```

TEMPLATE DÉPLOIEMENT K8S GÉNÉRIQUE

❖ TEMPLATE RÉUTILISABLE ET MONITORING INTÉGRÉ

Use Cases Supportés :

- ✓ Multi-ressources : Deployments, Services, Ingress, Policies, etc.
- ✓ Multi-environnements : DEV/STAGING/PROD via paramètres
- ✓ Multi-namespaces : Déploiement ciblé ou kube-system
- ✓ Comparaison état : Avant/Après déploiement automatique
- ✓ Logging avancé : Format JSON + parsing jq intégré
- ✓ Robustesse : Gestion d'erreurs avec continueOnError

KPI Mesurés :

- ✓ Temps d'installation : kubectl + kubelogin automatique
- ✓ État avant/après : Comparaison JSON automatique
- ✓ Flexibilité : 8 paramètres configurables
- ✓ Compatibilité : kubectl 1.32.3 + kubelogin v0.2.7
- ✓ Réutilisabilité : Template utilisé par X projets

AUTOMATISATION SCAN SÉCURITÉ TRIVY K8S

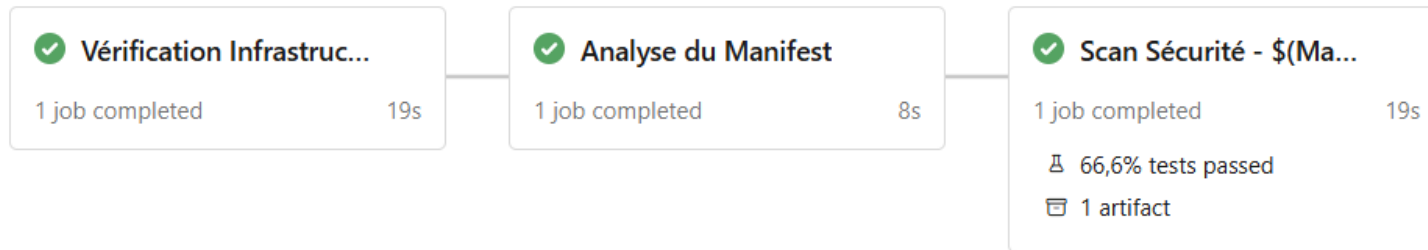
❖ Pipeline sécurité intégrée et dashboard visuel

Use Cases Supportés :

- ✓ Multi-environnements : DEV/STAGING/PROD
- ✓ Multi-manifests : nginx-test, api-test, monitoring, etc.
- ✓ Scan automatique : Config K8s + Images Docker
- ✓ Dashboard interactif : Métriques temps réel
- ✓ Rapports détaillés : JSON + JUnit + Artifacts
- ✓ Validation continue : Blocage optionnel sur vulnérabilités critiques

KPI Mesurés :

- ✓ Temps de scan : < 2min par manifest
- ✓ 41 vérifications : Sécurité K8s validées
- ✓ 7 vulnérabilités : Détectées et documentées
- ✓ 0 critiques : Aucune faille majeure
- ✓ Dashboard visuel : Intégré Azure DevOps Tests



```
# ===== SCAN SÉCURITÉ TRIVY =====  
- stage: SecurityScan  
  displayName: 'Scan Sécurité - $(ManifestName)'  
  dependsOn: AnalyzeManifest  
  jobs:  
    - template: azure-pipelines/integration-trivy.yml@pipeline_templates  
      parameters:  
        workingDirectory: '$(ManifestPath)'  
        scanTarget: 'config'  
        blockOnCritical: True
```

Syntaxe YAML :

```
Analyse du manifest: manifests/nginx-test.yaml  
Manifest trouvé: manifests/nginx-test.yaml  
Namespaces détectés: nginx-test
```

Syntaxe YAML valide

Artefact :

✓ 📄 trivy-security-report

📄 trivy-config.json

AUTOMATISATION INFRASTRUCTURE AKS

Pipeline Terraform modulaire et sécurisée

Use Cases Supportés :

- ✓ Multi-environnements : DEV/PRD/QAL avec variables dynamiques
- ✓ Clusters privés : DNS privé + réseau isolé
- ✓ Node pools multiples : System (management) + User (applications)
- ✓ Sécurité intégrée : SSH keys depuis Key Vault
- ✓ Approbation manuelle : Validation avant apply
- ✓ State management : Backend Azure Storage sécurisé

KPI Mesurés :

- ✓ 4 stages : Plan → Approbation → Apply → Outputs
- ✓ 2 modules AKS : Cluster principal + Node pool utilisateur
- ✓ 7 data sources : Récupération dynamique des ressources
- ✓ RBAC configuré : 2 groupes d'administration
- ✓ 6 outputs : Informations cluster exportées
- ✓ Convention nommage : \${Company}\${Language}-\${App}-\${Environment}

```
trigger: none
pr: none

parameters:
- name: Environment
  displayName: 'Environnement cible'
  type: string
  default: 'dev'
  values:
  - 'dev'
  - 'prd'
  - 'qal'

resources:
  repositories:
  - repository: pipeline_templates
    type: git
    name: Devops/terraform_cd
    ref: refs/heads/feature/aks_pipeline
```

❖ Variables Backend Dynamiques

```
ContainerName: tfstate
Key: ${FolderScope}/${ApplicationName}-${Environment}.tfstate
```

❖ Appel Module Azure AKS

```
# ===== Module Azure AKS =====
module "azure_aks" {
  source = "git::https://github.com/terraform-azure-modules/aks.git"

  # Variables obligatoires - Nommage dynamique
  name                        = "${var.Company}${var.Language}-${var.App}-${var.Environment}"
  resource_group_name        = "rg-${var.Company}${var.Language}-${var.App}-${var.Environment}"
  identity_resource_group_name = "rg-${var.Company}${var.Language}-${var.App}-${var.Environment}"
  resource_group_id           = "/subscriptions/${var.SubscriptionId}/resourceGroups/${var.ResourceGroup}"
  location                   = var.location

  # Ressources réseau dynamiques
  private_dns_zone_id         = var.private_cluster ? data.azure_rm_private_dns_zone["private_dns_zone_id"] : null
  vnet_subnet_id              = data.azure_rm_subnet["vnet_subnet_id"]

  # SSH Key sécurisée depuis Key Vault
  ssh_public_key              = data.azure_key_vault_certificate["ssh_public_key"]
}
```

Architecture déployée :

- ✓ Cluster AKS : Version K8s configurable, DNS privé
- ✓ Sécurité : SSH keys Key Vault, RBAC configuré
- ✓ Réseau : Subnet dédié, DNS service, network policy Azure
- ✓ Node Pools : System (management) + User (applications)
- ✓ Monitoring : Outputs pour intégration CI/CD

Templates utilisés :

- ✓ terraform-plan.yml : Validation et planification
- ✓ lead-approbation.yml : Approbation manuelle sécurisée
- ✓ terraform-apply.yml : Déploiement infrastructure
- ✓ terraform-outputs.yml : Export configurations cluster