Architecture Guide

Solution Overview

This Bicep solution follows a modular, application-centric architecture designed for:
- **Scalability**: Easy to add new applications and environments
- **Reusability**: Shared modules across all deployments
- **Maintainability**: Clear separation of concerns
- **Security**: Defense in depth with multiple security layers
- **Observability**: Comprehensive monitoring and logging

Design Principles

1. Modularity

Every component is a separate, reusable module:
- **Independence**: Modules can be updated independently
- **Testability**: Each module can be tested in isolation
- **Flexibility**: Mix and match modules as needed

2. Environment Separation

Each environment is isolated:
- **Resource Groups**: Separate RG per environment
- **Networks**: Isolated VNets per environment
- **Configurations**: Environment-specific parameters
- **Subscriptions**: Ready for multi-subscription deployment

3. Configuration as Code

All configuration is versioned:
- **Parameter Files**: `.bicepparam` files for each environment
- **Token Files**: Environment-specific tokens in JSON
- **Common Config**: Shared configuration in `common.json`
- **No Manual Changes**: Everything defined in code

Architecture Layers

Layer 1: Configuration

```
config/
├── naming/           # Naming convention templates
├── tags/             # Tagging templates
├── tokens/           # Environment-specific settings
└── common.json       # Shared configuration
```

Purpose: Centralized configuration management

Key Features:
- Consistent naming across all resources

- Standardized tagging for governance
- Environment-specific overrides
- Common settings shared across environments

Layer 2: Reusable Modules

```
modules/
├── compute/          # VM, scale sets
├── network/          # VNet, NSG, ASG
├── storage/          # Storage accounts
├── monitoring/       # Log Analytics, diagnostics
├── backup/           # Recovery Services
├── security/         # Managed Identity
└── keyvault/         # Key Vault
```

Purpose: Building blocks for infrastructure

Key Features:
- Parameterized templates
- Comprehensive documentation
- Input validation with decorators
- Rich outputs for module chaining

Layer 3: Application Orchestration

```
applications/
└── step/
    ├── main.bicep       # Orchestration template
    ├── dev/             # Dev environment
    ├── test/            # Test environment
    ├── uat/             # UAT environment
    └── prod/            # Production environment
```

Purpose: Application-specific infrastructure

Key Features:
- Composes modules into complete solutions
- Environment-specific parameter files
- Application-level configuration
- Manages resource dependencies

Network Architecture

Hub-Spoke Ready Design

While this implementation shows individual VNets per application, it’s designed to easily extend to hub-spoke:

```
      [Hub VNet]
      /    |    \
     /     |     \
[App1] [App2] [App3]
 VNet   VNet   VNet
```

Current Setup (Per Application):

```
Virtual Network (10.x.0.0/16)
  └── Application Subnet (10.x.1.0/24)
      ├── Network Security Group
      ├── Service Endpoints (Storage, KeyVault)
      └── Application Security Groups
```

Address Space by Environment:
- Dev: 10.0.0.0/16
- Test: 10.1.0.0/16
- UAT: 10.2.0.0/16
- Prod: 10.3.0.0/16

Network Security

Defense in Depth:
1. **NSG (Network Security Group)**
- Stateful firewall at subnet level
- Inbound and outbound rules
- Priority-based rule processing

1. **ASG (Application Security Group)**
   - Logical grouping of VMs
   - Simplifies NSG rule management
   - Enables application-centric security

2. **Service Endpoints**
   - Secure access to Azure services
   - Traffic stays on Azure backbone
   - No public IP exposure needed

Compute Architecture

Virtual Machine Design

Components:

```
Virtual Machine
├── OS Disk (Premium/Standard SSD)
├── Data Disks (0-64 disks)
├── Network Interface
│   ├── Private IP (static/dynamic)
│   └── Public IP (optional)
├── Managed Identity
│   ├── System Assigned
│   └── User Assigned (optional)
├── Boot Diagnostics
└── Azure Monitor Agent
```

Scaling Strategy:
- **Dev**: Small VMs (B-series), 1 instance
- **Test**: Medium VMs (D-series), 2 instances

- **UAT**: Large VMs (D-series), 2+ instances
- **Prod**: XL VMs (D/E-series), 3+ instances, Availability Zones

Storage Architecture

Storage Account Design

Boot Diagnostics Storage:
- One per resource group
- Standard_LRS (locally redundant)
- Secure by default (HTTPS only, TLS 1.2)
- Network-restricted in prod/UAT

Data Disks:
- Attached to VMs as needed
- Premium_LRS for production workloads
- StandardSSD_LRS for dev/test
- Configurable per environment

Monitoring Architecture

Log Analytics Workspace

Centralized Logging:

```
Log Analytics Workspace
  ├── VM Logs
  ├── Performance Metrics
  ├── Security Events
  └── Custom Logs
```

Data Collection:
- Azure Monitor Agent on each VM
- Boot diagnostics in storage account
- NSG flow logs (can be enabled)
- Activity logs from Azure platform

Retention Strategy:
- Dev: 30 days
- Test: 60 days
- UAT: 90 days
- Prod: 180 days

Security Architecture

Identity and Access Management

Managed Identities:

```
VM with System Assigned Identity
  ├── Automatic lifecycle management
  ├── No credential management needed
  └── RBAC assignments for Azure resources
```

Key Vault Integration:

```
Key Vault
  ├── Secrets (passwords, connection strings)
  ├── Keys (encryption keys)
  └── Certificates (SSL/TLS certs)

  Access via:
  ├── VM Managed Identity
  ├── RBAC (Key Vault Secrets User)
  └── Service Endpoints
```

Network Security Model

Zero Trust Approach:
1. **Default Deny**: All traffic denied by default
2. **Explicit Allow**: Only required traffic allowed
3. **Least Privilege**: Minimum necessary access
4. **Segment**: Network segmentation with NSGs/ASGs

Backup and Disaster Recovery

Backup Architecture

```
Recovery Services Vault
  ├── Backup Policy
  │   ├── Daily: 2:00 AM UTC
  │   ├── Retention: 30 days (daily)
  │   ├── Retention: 12 weeks (weekly)
  │   └── Retention: 12 months (monthly)
  └── Protected VMs
```

Environment Strategy:
- Dev: No backup (saves cost)
- Test: Basic backup (30 days)
- UAT: Extended backup (60 days)
- Prod: Full backup with long retention

Naming Convention

Resource Naming Pattern

```
{resourceType}-{applicationName}-{environment}-{region}-{instance}
```

Examples:
- Resource Group: `rg-step-dev-eus`
- Virtual Machine: `vm-step-dev-eus-001`

- VNet: `vnet-step-dev-eus`
- Storage: `stepdeveus001` (no dashes)
- Key Vault: `kv-stepdeveus` (max 24 chars)

Benefits:
- Easy identification
- Consistent across environments
- Supports automation
- Facilitates cost tracking

Tagging Strategy

Standard Tags

Every resource gets these tags:

```
{
  "Application": "step",
  "Environment": "dev",
  "ManagedBy": "Bicep",
  "DeploymentDate": "2026-01-08",
  "CostCenter": "IT-Dev",
  "Owner": "devteam@example.com",
  "Department": "Engineering"
}
```

Use Cases:
- Cost allocation and chargeback
- Resource governance and compliance
- Automation and orchestration
- Documentation and inventory

Deployment Architecture

Deployment Scope

```
Subscription Scope Deployment
  └── Creates Resource Group
      └── Deploys Resources
          ├── Network
          ├── Compute
          ├── Storage
          ├── Monitoring
          ├── Backup
          └── Security
```

Multi-Subscription Support

Scenario 1: Single Subscription

```
Subscription
  ├── RG-App1-Dev
  ├── RG-App1-Test
  ├── RG-App1-UAT
  └── RG-App1-Prod
```

Scenario 2: Multi-Subscription (Future)

```
Sub-NonProd           Sub-Prod
  ├── RG-App1-Dev      ├── RG-App1-Prod
  ├── RG-App1-Test     └── RG-App1-DR
  └── RG-App1-UAT
```

Extensibility

Adding New Components

To add a new module:
1. Create module in `/modules/{category}/`
2. Define parameters and outputs
3. Add comprehensive documentation
4. Reference in application orchestration

To add a new application:
1. Create folder in `/applications/{appname}/`
2. Copy and customize `main.bicep`
3. Create environment-specific `.bicepparam` files
4. Deploy and test

Integration Points

External Services:
- Azure AD for authentication
- Azure DevOps for CI/CD
- Azure Monitor for alerting
- Microsoft Defender for security
- Azure Policy for governance

On-Premises Connectivity:
- VPN Gateway (can be added)
- ExpressRoute (can be added)
- Azure Bastion (can be added)

Performance Considerations

VM Sizing

CPU/Memory Ratio:
- **General Purpose** (D-series): 1:4 ratio
- **Memory Optimized** (E-series): 1:8 ratio
- **Compute Optimized** (F-series): 1:2 ratio

Storage Performance

Disk Types:
- **Premium SSD**: 5,000+ IOPS, < 1ms latency
- **Standard SSD**: 500+ IOPS, < 10ms latency
- **Standard HDD**: 500 IOPS, ~10-20ms latency

Network Performance

Accelerated Networking:
- Enable for D-series and above
- Up to 30 Gbps network throughput
- Single root I/O virtualization (SR-IOV)

Cost Optimization

Right-Sizing Strategy

1. **Start Small**: Begin with smaller VMs
2. **Monitor**: Use Azure Monitor to track utilization
3. **Adjust**: Resize based on actual usage
4. **Reserved Instances**: For long-running prod workloads

Cost-Saving Features

- **Auto-shutdown**: Schedule VM shutdown (dev/test)
- **Spot VMs**: For fault-tolerant workloads
- **Hybrid Benefit**: For Windows Server licenses
- **B-series**: For low-utilization workloads

Next: See DEPLOYMENT.md (./DEPLOYMENT.md) for deployment instructions