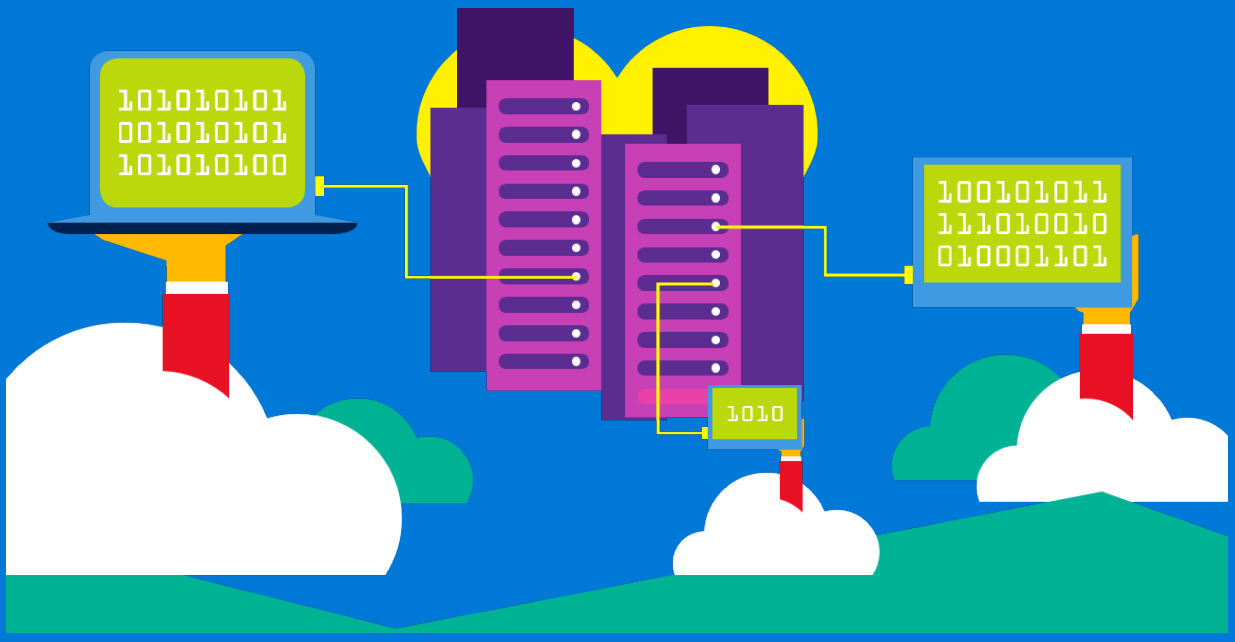
Microsoft



Enterprise cloud control plane planning

End-to-end pipelines for automating Microsoft Azure deployments

By Tim Ehlen
AzureCAT

October 2018

Contents

Authored by Tim Ehlen. Edited by Nanette Ray. Reviewed by AzureCAT.

Overview

Imagine a fully automated, end-to-end pipeline for your cloud deployments—one that encompasses and automates *everything*:

- Source code repos.
- The build and release iterations.
- Agile processes supported by continuous integration and continuous deployment (CI/CD).
- Security and governance.
- Business unit chargebacks.
- Support and maintenance.

Azure services and infrastructure-as-code (IaC) make control plane automation very achievable.

Many enterprise IT groups dream of creating or unifying their disparate automation processes and supporting a common, enterprise-wide datacenter control plane in the cloud that is integrated with their existing or new DevOps workflows. Their development environments may use Jenkins, Azure DevOps Services (formerly Visual Studio Team Services), Visual Studio Team Foundation Server (TFS), Atlassian, or other services. The challenge is to automate beyond the CI/CD pipeline to the management and policy layers. From a planning and architecture standpoint, it can seem like an overwhelming program of interdependent systems and processes.

This guide outlines a planning process that you can use for automated support of your cloud deployments and DevOps workflows beyond the CI/CD pipeline. The Azure platform provides services you can use, or you can choose to work with third-party or open source options. The process is based on real-world examples that we have deployed with enterprise customers on Azure.

Tracking the program step by step

An automated enterprise cloud deployment system on Azure may seem daunting, particularly for development teams that use multiple source code repositories. However, you can trace each step in the workflow execution as the following figure shows. Each component in this workflow can be architected, implemented, and integrated using agile methods into a cohesive, well-organized whole that supports integration and collaboration across all the areas of concern. This workflow applies whether you're using IaaS or PaaS on Azure, and whether your team writes your code or a vendor does it.

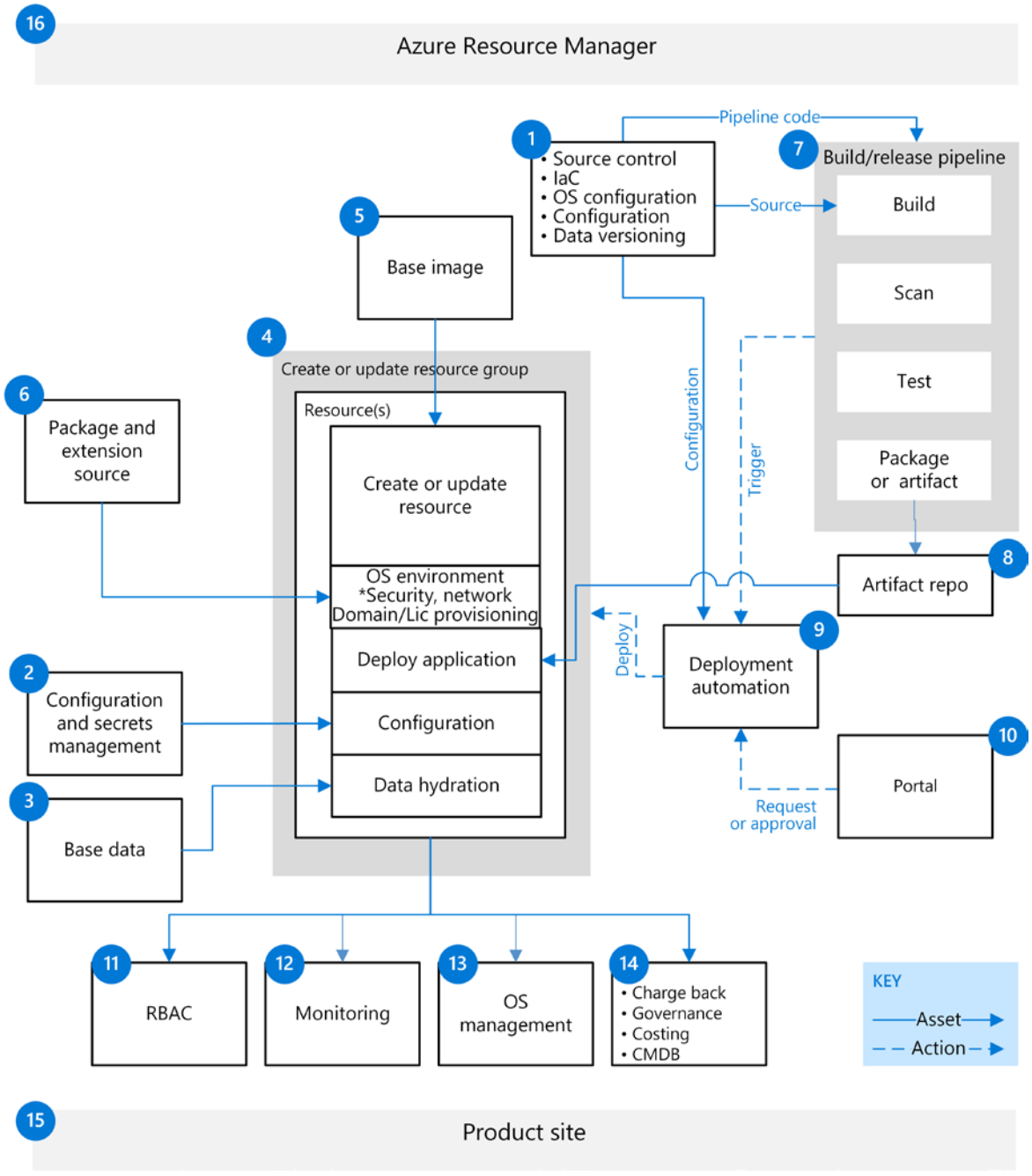


Figure 1. End-to-end workflow for control plane automation of a solution deployed on Azure.

At the top of Figure 1, Azure Resource Manager and the Azure back plane manage and maintain all the resources. At the bottom is the project site, product home, or other tracking place for your particular product or workload. In between is the build and release pipeline, source control, secrets management, deployment orchestration, and all the steps of the CI/CD pipeline. The workflows in Figure 1 differentiate the management of resources and assets (shown with solid blue lines) from the actions taken by automation scripts (shown with dashed blue lines).

1 Source control

Inputs	Outputs
External action—for example, developer checks in code	Source code Pipeline as code Deployment IaC Data versioning

Automated IaC starts with a source control repository. Source control in modern pipelines can contain the vital components of your automation.

Source code

The application source is the most obvious component for application automation, but many enterprise IT organizations have yet to modernize or even use their source control. Consolidation of disparate source systems or deprecated systems can be a major effort. Considerations for migration include the following:

- Consolidation of enterprise code.
- Integration with automation pipelines and work tracking.
- Migration from legacy systems or storage.
- Selection of a management system such as Git management.
- An organizational hierarchy for the enterprise portfolio.
- Training to source control systems such as Git.

Automation pipeline

Modern build and release systems such as Azure DevOps and Jenkins support automation through pipeline as code. Each step in the build and release pipeline is described in a file like any other piece of code and stored in a source control repository. These automation pipeline systems can make use of customized or shared build pipelines from source control, greatly increasing the flexibility and reusability of pipeline code.

Deployment configuration

Source control can specify the environment where an application is deployed. An automated deployment configuration can include:

- Infrastructure as code (IoC).
- Deployment automation.

- Environment configuration (no secrets in source control).
- Data hydration and environment validation automation.

Tools you can use

Azure DevOps Services, GitHub, TFS

2 Configuration and secrets management

Inputs	Outputs
External action	Environment configuration
	Secrets

To provide security and governance controls for environmental settings, the workflow must include a way to manage configurations and secrets. Configuration settings can be transient, persisting only as long as an environment, or they may need to last through deployments of a particular environment. Configuration settings for environments are stored in a project's source control, or they may be stored independently in a configuration management database (CMDB) as with charge back and support details, for example.

Secrets should always be stored in a secure mechanism. Secrets that get reused require special treatment. For example, details about connections to external systems or certificates need to persist through the destruction and construction of environments. A service such as Azure Key Vault is ideal.

Tools you can use

Key Vault, Service Now, CMDB

3 Base data

Inputs	Outputs
External action	Based data

An end-to-end pipeline often requires external systems of base data that are used to recreate environments or used as a backup or intermediate step when migrating applications to the cloud. When an automated pipeline can automatically restore base data, you can quickly reproduce specific conditions for a test environment, set up new scenarios or preproduction validation for testing, and provide cleansed data for recreating production environments.

Tools you can use

Azure Storage, BAKPAK files, various database systems

4 Deployment phases

Inputs	Outputs
Automation commands	Resources
	OS environment
	Application deployment
	Configuration
	Data hydration

Whether a cloud deployment includes IaaS or PaaS, or both, the following phases apply:

- Create and update resource groups.
- Deploy and provision the OS environment (IaaS only).
- Deploy the application.
- Configure the resources.
- Load data (that is, data hydration).

It's vital to organize the workflow design correctly for these deployment phases. Everything in Azure revolves around the subscription and the resource group or groups that contain a service. Many organizations find it helpful to organize resource groups according to the type of resource being deployed, such as virtual machines, storage, and databases. Others organize the resources based on the workload being deployed, grouping according to application, lifespan, or ownership and control.

In any case, when designing an organization scheme for resource groups, consider the following:

- Role-based access control (RBAC) provides security controls over the resources in a resource group. (For details, see 11 RBAC later in this guide.)
- Governance, monitoring, and maintenance of the resources in the resource group.
- Cost capture and billing of the resources based on the organization of Azure subscriptions.
- Life cycle for a resource group and its relationship to the resources it contains.

Tools you can use

Azure Resource Manager, PowerShell, CLI

5 Base image

Inputs	Outputs
Image automation	Virtual machine images

Management, organization, and automation of base images used in IaaS scenarios can be simple or very intricate. When designing a base image system, the goal is strike a balance among all the variables: optimization, control, governance, cost, availability, and simplicity. Many enterprises choose to build IaaS systems from stock Azure Marketplace images, and then build up a base deployment using Desired State Configuration (DSC) or another type of deployment automation. Other workloads require custom base images built for rapid deployment. In a modern cloud environment, custom image management can become a large pipeline system that incorporates multiple OS types, versions, and roles across global regions of deployment. The larger the pipeline, the more important it is to balance all the requirements before choosing a management system.

Tools you can use

Azure Marketplace, Shared Image Gallery, Storage

6 Package and extension source

Inputs	Outputs
External action	Library packages

Deployment of libraries, modules, or agents on Azure compute systems can be as simple as providing selective Internet access to package repositories. For a more restrictive design, complete library management systems can be used, such as internal NuGet repositories or node package manager (NPM) systems. The package and extension source may require one or many systems, and the workflow includes their management, networking requirements, and RBAC control design.

Tools you can use

NuGet, Artifactory, Azure DevOps Services, NPM

7 Build and release pipeline

Inputs	Outputs
Pipeline as code	Artifacts
Source code	Log
	Deployment certifications

For the purposes of this guide, the build and release pipeline specifically refers to building source

control code and configuration. How this is done depends on the tool used, but the standard pipeline orchestration consists of these tasks:

- Build
- Test
- Verify
- Publish artifact
- Orchestrate or trigger environment deployments

The automation systems used for this pipeline may orchestrate the deployment of a specific cloud environment service, or it may just call the service's built-in deployment orchestration. For this reason, the deployment operations (shown as steps 7 and 9 in Figure 1) are separated. One system may be used for custom-built deployments, and others to handle prepackaged workloads or deployments based on IaaS, containers, or PaaS.

Tools you can use

Azure DevOps Services, Jenkins

8 Artifact repository

Inputs	Outputs
Artifacts	Deployment packages

Build and release systems typically persist their output in a repository for retrieval later by deployment systems. To manage these operations, Maven, NuGet, Storage, or another management system is required. Today's systems support many protocols and optimize management tasks. The network design must provide appropriate access to the deploying systems, orchestration agents, and cloud services.

Tools you can use

Azure DevOps Services, Artifactory, Nexus

9 Deployment automation

Inputs	Outputs
IaC	Deployments
Packages	
Deployment triggers	

One of the most essential pieces of your pipeline is the system used for deployment automation of artifacts and IaC to cloud services. Deployment automation systems are the key control point for your cloud deployments and environment. These systems vary in design from simple to complex as the following examples show:

- Scripted or plugin deployment from build and release agents.
- Azure Resource Manager deployments using automation agents.
- Automation systems based on domain-specific languages such as Ansible and Chef.
- Orchestration systems for automation agents and credential management.

One major consideration when designing this system is RBAC and the proper management of the rights needed to deploy services to the cloud. What rights are needed to create, delete, or modify deployments inside or outside a resource group? Often, one set of rights is required to create a resource group in Azure and another to deploy services in the resource group, for example.

Tools you can use

Azure Resource Manager, Azure DevOps Services, Jenkins, Ansible, Terraform, Chef

10 Portal

Inputs	Outputs
External action	Automation triggers

Enterprise governance requires standards of access to IT systems, configuration, and work. A standardized portal is commonly used for this purpose. A business unit uses the portal to request new services or work. The request can then be integrated into a CMDB and other systems to provide visibility into the IT landscape. As the primary or sole point of access for service requests, a portal often forms the start of the pipeline in a hybrid cloud environment.

Tools you can use

ServiceNow, BMC

11 RBAC

Inputs	Outputs
Rights management	Access control
Application registrations	

An Azure Active Directory(Azure AD) service principal is needed to define the policy and permissions for each role authorization in the workflow via RBAC. Service principals grant authenticated, authorized access to Azure resources. RBAC defines and controls access to Azure resources for users or service principals through their assigned role.

The basis of RBAC is your identity solution. Usually this workflow starts with Azure AD, synchronized to an enterprise Active Directory domain. Conformance with your local identity policies is essential to determine compatibility, and these policies must be assigned to service principal accounts in Azure AD for system-to-system communications.

The transition to the cloud requires organizations to rethink their old perimeter-based security model and their reliance on its assumptions. Many security policies must be rethought. For

example, a simple web application hosted on a private network trusts the perimeter security to protect a non-SSL anonymous website. When moving to the cloud, it can be easy to just say it needs to be on a private network and thus incur significant cost for an application of low business value. Adding SSL and authentication can be another low-effort process to maintain high impact with minimal modernization. Azure App Service offers a built-in authorization and authentication feature nicknamed Easy Auth that simplifies authentication to Azure AD but otherwise allows access to the application from a public endpoint, assuming routing of IP traffic over a private ExpressRoute circuit.

Tools you can use

Azure AD

12 Monitoring

Inputs	Outputs
Events	Events
Telemetry	Alerts
Logs	

Monitoring services monitor the resources created and managed through automation and can notify you of CPU utilization, virtual machine status, and other key performance indicators. The workflow design must consider automation and alerting for each service across the portfolio and can use shared services, an application team, or both. Security policies and governance are also tracked in these systems.

Tools you can use

Azure Security Center, Azure Monitor, Azure Application Insights, Azure Log Analytics

13 OS management

Inputs	Outputs
Agent registrations	Log Analytics

Management of Linux and Windows operating systems for IaaS compute resources can be greatly aided by Azure Cloud Services. The automation design can even utilize existing on-premises processes or all-new, cloud-based systems in Azure Update Management or Azure Automation plus Log Analytics. In this area of the automation pipeline, considerations include how to manage updates and base images and how to recreate services.

For virtual machines and other IaaS components, you can use Azure Update Management or similar agents and services or existing systems. Many organizations use automation initiatives as an opportunity to rethink their OS management systems.

Tools you can use

Azure Monitor, Update Management or Azure Automation + Log Analytics, Chef

14 Chargeback, governance, costing, CMDB

Inputs	Outputs
Resource tags	Configuration

The overall governance and cost recovery of the systems deployed in your organization should be considered at an early stage. You must collect thorough requirements from all the organizations or business units about cloud billing and cost recovery. Consider whether billing needs to occur at the enterprise level, and then pass down to business units, or whether billing starts at the business unit.

Tools you can use

Azure Billing API, Azure Resource Manager, Azure Cost Management, BMC Remedy, ServiceNow

15 Product site

Inputs	Outputs
	Documentation
	Tracking

Most applications or workloads publish a site for management and support information. One example is an Azure DevOps project site with source, work tracking, documentation, and release tracking. Another example is an IT system for tracking all support, changes, and tickets performed on a site. This product site can contain all necessary links and actions required to support the software development life cycle for the application or workload.

Tools you can use

Azure DevOps Services, ServiceNow, BMC Remedy

16 Azure Resource Manager

Inputs	Outputs
Azure configuration	Subscription
	Resource group
	Networking
	Other shared services

All Azure resources are contained inside a tenant and subscription within your Azure account. Deployments must abide by the policies, RBAC security, and organizational structure prescribed by the account.

The high-level architecture of the workflow associated with Azure configuration and setup is composed of:

- Subscription and management group model
- RBAC security model
- Networking design
- Policies and security
- Naming conventions
- Limits management
- Shared services
- Identity

Areas of responsibility

The design and workflow of an end-to-end pipeline breaks down into several distinct areas or responsibilities. These areas closely match current IT specialties as Figure 2 shows.

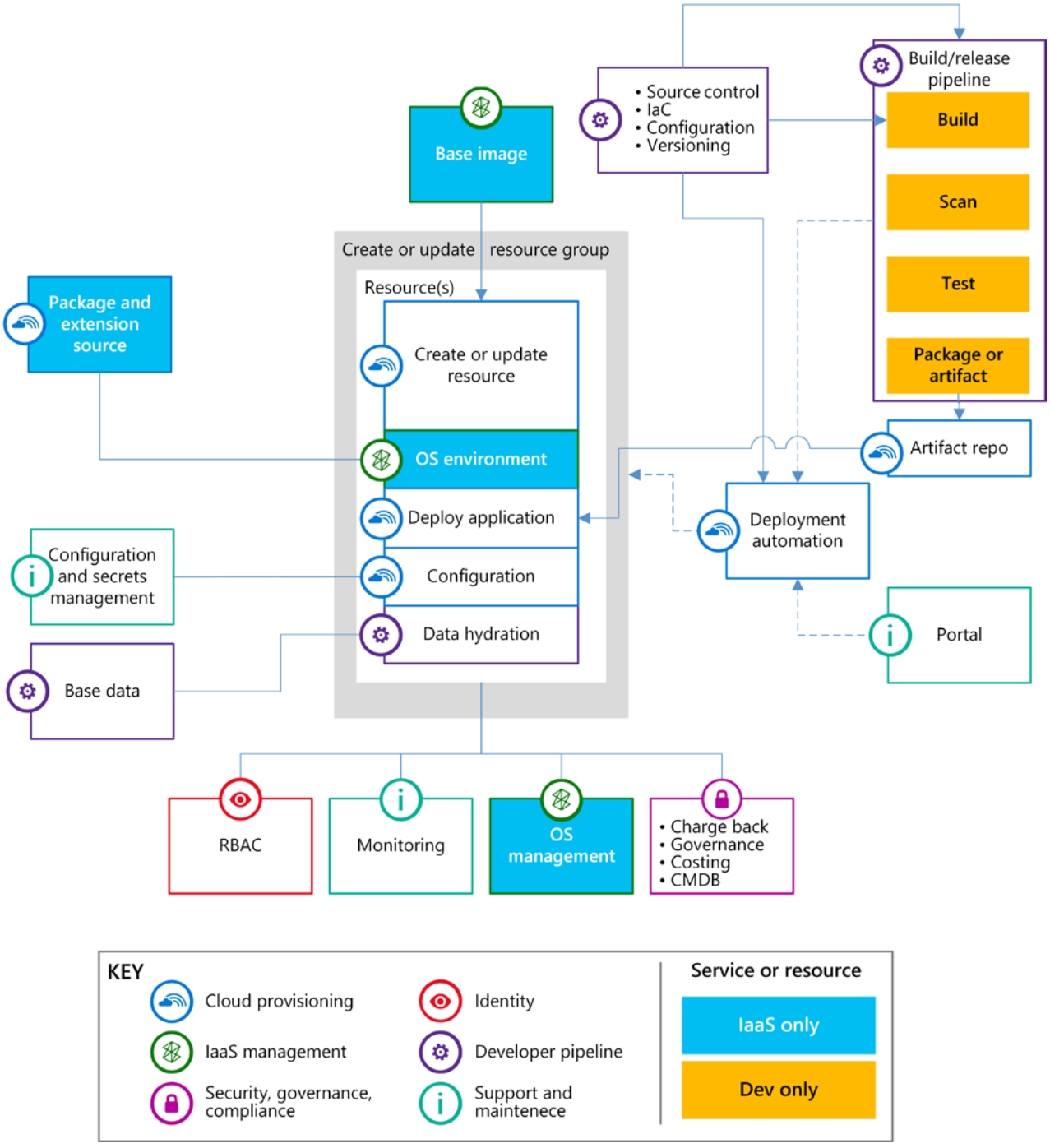


Figure 2. The key shows the areas of responsibility for various legs of the end-to-end pipeline.

Cloud provisioning

The team responsible for cloud provisioning deploys services and binaries into the cloud environment. Services can encompass PaaS and IaaS offerings, and the team is responsible for the automation used to deploy the needed functionality and services. For these resources, the team's responsibilities could include management of the deployment orchestration systems all the way to managing shared libraries.

As you get started with cloud provisioning, consider assigning to your current datacenter automation teams who specialize in PowerShell, Ansible, or Chef.

IaaS management

At the outset, this responsibility is typically handled by the current OS maintenance teams from your datacenter. Your organization may continue to use some current systems or transition services directly to their cloud counterparts.

Security, governance, and compliance

This area of responsibility is often staffed by existing risk, governance, and information security teams. A cloud migration or adoption initiative can quickly strain these teams as an on rush of new cloud services across various deployment patterns requires a new paradigm for security assessments and risk acceptance.

Consider using outside expertise to help realign your security, governance, and compliance requirements to an appropriate cloud pattern.

Identity

Most organizations already have teams in place to handle identity requirements such as creating the Azure AD tenant and connecting the on-premises Active Directory to Azure AD. To support IaaS deployments, additional Active Directory servers in the cloud are needed. This area of responsibility is critical when your organization moves to a RBAC-based security model.

Development pipeline

DevOps, agile development, and CI/CD are essential elements of a modern software development life cycle. The systems that support these functions are what is normally thought of in a development pipeline. Your current development automation experts will gravitate to these services.

Support and maintenance

Support and maintenance tasks typically fall under the jurisdiction of your monitoring organization. This team is responsible for monitoring policy compliance, system monitoring, intrusion and security monitoring in the cloud. The goal is a design that enables the team to monitor from a single pane of glass, such as a management console, or to provide a full, 360-degree view of the hybrid cloud landscape.

Next steps

End-to-end automation pipelines are more than a pipe dream. Moving to the cloud presents many design challenges but also makes increasing levels of automation not only a real possibility but a necessity. As this guide explains, the cloud control plane consists of specific components, workflows, roles, and responsibilities. With an understanding of the system requirements, your organization can organize the program and start planning beyond the CI/CD pipeline on Azure.

To help you get started, your cloud program team can begin to enumerate your organization's workstreams and team assignments using a table like this one.

ID	System	Team	Wiki	Status	Notes
1	Source control	T1	http://..	Green	
2	Configuration and secrets	T2	http://..	Green	
3	Base data	T3	http://..	Green	
4	Deployment phases	T4	http://..	Green	
5	Image management	T5	http://..	Green	
6	Package and extensions	T6	http://..	Yellow	Determining solution
7	Build pipeline	T7	http://..	Green	
8	Artifacts	T8	http://..	Green	
9	Deploy automation	T9	http://..	Green	
10	IT portal	T10	http://..	Green	
11	RBAC/identity	T11	http://..	Green	
12	Monitoring	T12	http://..	Green	
13	OS management	T13	http://..	Green	
14	Governance	T14	http://..	Green	
15	Product site	T15	http://..	Green	
16	Azure configuration	T16	http://..	Green	

This list will change over time as you iterate deeper into designs and implementations. Some areas will be simplified and others strengthened as your organizational needs change. For example, some organizations have deep compliance or security requirements; some may have little to no custom development, and others may deliver commercial services to customers. Regardless of the requirements and unique conditions for your organization, this generic starting point is one mechanism to start architecting and planning your enterprise control plane.

Learn more

Azure DevOps documentation

Azure Resource Manager overview

Manage access using RBAC and Azure Resource Manager templates