**Introduction to Resource Management**

This module is about building automation to provision databases in Azure Pipelines, by using Azure Resource Manager templates and Azure SQL Database as an example.

In this module, you use *infrastructure as code* to deploy the *Space Game* web application that's using a SQL database. You deploy your infrastructure by using an Azure Resource Manager template and a multistage pipeline with Azure Pipelines. This pipeline will also manage changes to the database schema. You use Azure Key Vault to store secrets and later access those secrets from the Azure Resource Manager template and Azure Pipelines.

In Manage database changes in Azure Pipelines , you added stages to your pipeline. These stages checked for database schema changes, waited for DBA approval of those changes, and then applied the changes to the database.

However, you created the SQL Server logical instance, the SQL database, and the Azure App Service infrastructure manually. Here, you automate the infrastructure creation from a stage in Azure Pipelines.

**Learning objectives**

In this module, you:

* Deploy your infrastructure by using an Azure Resource Manager template in Azure Pipelines.
* Create your Azure SQL database with data from a *.bacpac* file in your Azure Resource Manager template.
* Use Azure Key Vault to store and access secrets from Azure Pipelines and your Azure Resource Manager template.

**Prerequisites**

The modules in this learning path form a progression.

To follow the progression from the beginning, be sure to first complete these learning paths:

* Evolve your DevOps practices
* Build applications with Azure DevOps
* Deploy applications with Azure DevOps

This module builds on Manage database changes in Azure Pipelines , from the Deploy applications with Azure DevOps  learning path. In that module, you provisioned infrastructure manually. In this module, you build an Azure Resource Manager template that produces the same result through automation.

We also recommend that you start at the beginning of this learning path: Automate your deployments with Azure DevOps .

If you want to go through just this module, you need to set up a development environment on your Windows, macOS, or Linux system. You need:

* An Azure DevOps organization .
* An Azure subscription .
* A GitHub  account.
* Visual Studio Code .
* .NET Core 3.1 SDK .
* Git .

You can get started with Azure and Azure DevOps for free.

This environment lets you complete the exercises in this and future modules. You can also use it to apply your new skills to your own projects.

**Note**

Keep in mind that you can use Azure DevOps to build and deploy almost any kind of application written in any language. In this module, you'll be working with a .NET Core application written in C#.

You don't need to be an expert in .NET or C# to complete this module. You can apply the patterns you learn here to your own projects that use your favorite programming languages and frameworks.

# Automate infrastructure deployment by using Azure Resource Manager templates

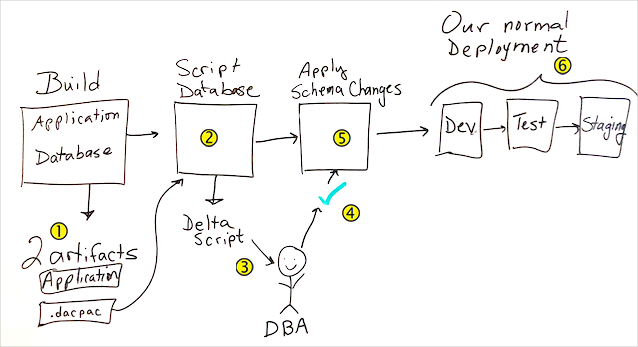
* 8 minutes

The team wants to try using an Azure Resource Manager template to automate the deployment of the Space Game web application. This automation includes the SQL Server logical instance and the SQL database, as well as the Azure App Service plan and the App Service instance.

The web app is still in development, and database schema changes will happen. The database administrator (DBA) is responsible for the integrity of the database and needs to approve any changes made to the schema before they're applied.

In the Manage database changes in Azure Pipelines  module, you created a pipeline with a stage that builds the .NET Core web app project and the SQL Server Data Tools database project. Then, you added a stage to discover changes to the database schema. The next stage that you added applied those schema changes if the DBA approved them. Then the web app was deployed.

As a reminder, here's the pipeline diagram from that previous module.



The Build stage has a job to build the web app and a job to build the database project. Call-out 1 on the whiteboard. The database project produces a .dacpac file that is a build artifact.

The next stage Call-out 2 on the whiteboard. scripts the database changes so that the Call-out 3 on the whiteboard. DBA can verify the changes before they're applied.

An Call-out 4 on the whiteboard. approval is added to the next stage that Call-out 5 on the whiteboard. applies the database changes. The last three stages Call-out 6 on the whiteboard.  deploy to Dev, Test, and Staging environments.

In that module, you created the infrastructure manually before you ran the pipeline. The Dev, Test, and Staging stages all had identical infrastructure and shared a common database. As the web application project matures, the infrastructure needs for each of these stages also changes. The team is looking at Resource Manager templates to automate the deployment process in each stage.

In this section, you:

* Learn about infrastructure as code using Resource Manager templates.
* Design a Resource Manager template for Azure App Service and Azure SQL Database to use in Azure Pipelines.
* Understand how to use Azure Key Vault to store and retrieve secrets by using Resource Manager templates and Azure Pipelines.
* Learn what you need to add a Resource Manager template to Azure Pipelines.

Let's listen in on the morning meeting.

## The meeting

Andy has called the team together to discuss planning for the infrastructure changes that are the result of a maturing project.

**Andy:** Good morning everyone. We're getting closer to our release date, and Tim and Amita have some concerns they want to talk about. Amita?

**Amita:** Tim and I have been talking about the current flow for our release, and we believe it's time to push toward infrastructure that goes beyond development and is more like what the final product will need. For example, I need to test on machines with higher throughput and better redundancy. And staging needs to have data that's more real-world. Tim is concerned that this will slow down our release flow, since he'll have to provision all of these stages differently and maintain them. We're concerned how this will affect our pipeline.

**Mara:** I think I can help here. We've been trying out infrastructure as code with Terraform. Remember, infrastructure as code allows us to describe the infrastructure we need through code. This way, we can maintain the application code and everything we need to deploy the application in our central code repository. It has been working well for us, but I've been wanting to try Azure Resource Manager templates. This might be a good time to try one as proof of concept. We could create a template that provides parameters at each stage so that each stage can be provisioned with what it needs. The pipeline can run the template and use a specific parameter file for the stage it's provisioning.

**Tim:** I was hoping we could do that. It sounds like automation would enable us to more easily provision and maintain a separate database at each stage of the pipeline. But what are Resource Manager templates?

### Use Resource Manager templates for infrastructure as code

Recall that Resource Manager is the interface for managing and organizing cloud resources. A Resource Manager template defines all the resources in a deployment, so you can deploy the template into a resource group as a single operation. We provide a link to a module on Resource Manager templates at the end of this module.

**Mara:** Resource Manager templates are JSON files that define all the resources we need so that Resource Manager can deploy the resources. We can get a quickstart template to start with from the quickstart template gallery . These were created by the community with common infrastructure and would be a good place to start if we didn't already have infrastructure in place. Since we have infrastructure now, we can export the entire resource group as a template from the Azure portal . From there, we can modify the template to make it more reusable across the pipeline.

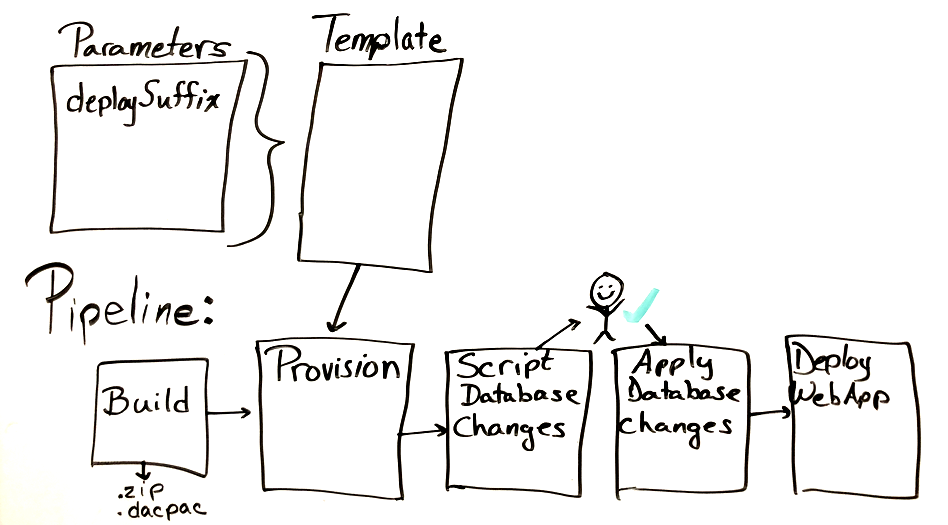
**Andy:** Let's start drawing out our plan.

Andy moves to the whiteboard.

### Design a Resource Manager template for App Service and SQL Database to use in Azure Pipelines

The template needs to have information passed in as parameters to have the reusability that the team needs. It will also need to output information from the created resources. For example, it might output the database connection string to use later in the pipeline. Let's listen in as the team decides what it will need in the template.

**Andy:** We need to create template and parameter files and change the pipeline to use them. The only parameter so far is the resource name suffix that represents the stage we're deploying to. For example, we're using dev for the development stage.

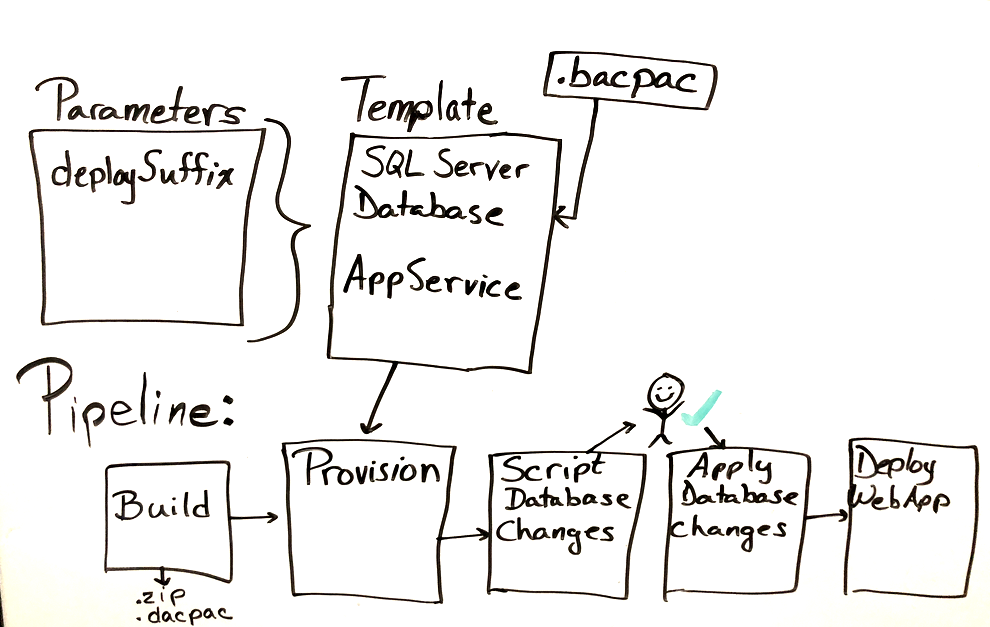


What goes in the template?

Mara pulls out her laptop and does a quick search.

**Mara:** Everything we had to create manually the last time we worked with the database changes in the pipeline can now go in the template file. We can even deploy the database by using the .bacpac file. So that means we'll provision the SQL Server instance, the SQL database, the App Service plan, and the App Service instance. I propose we create just the infrastructure we need for one stage. Let's start with the Dev stage. Then we can reuse this template for the other stages; we just need to change the parameters we give the template.

**Andy:** Great. I'll draw that in.



**Tim:** But what about the administrator password for the SQL Server instance? I don't want that information to be stored in our repository.

Andy does some typing on his laptop.

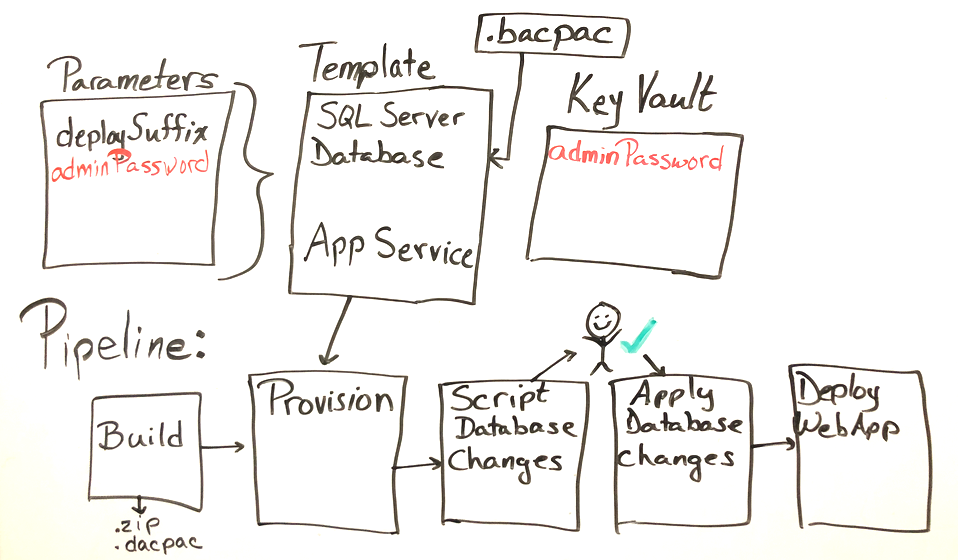
### Use Azure Key Vault to store and retrieve secrets by using Resource Manager templates and Azure Pipelines

Azure Key Vault enables you to securely store secrets. For example, you can store your API keys and passwords or anything that needs tight control over who has access to it.

**Andy:** It looks like we can store the administrator password in Azure Key Vault. The template parameter file can read from Key Vault.

Andy draws on the whiteboard.

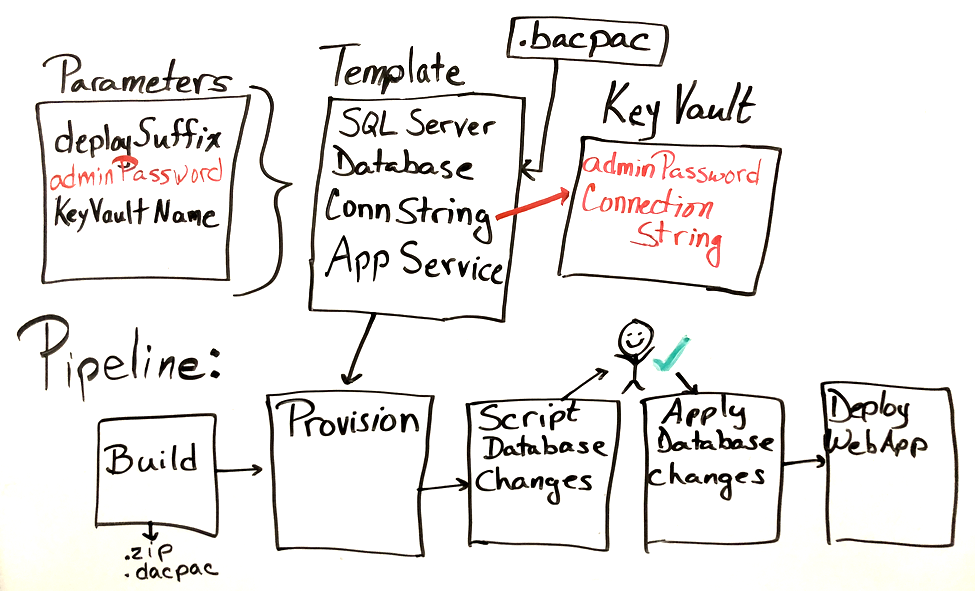
So now we have Key Vault and another parameter.



**Tim:** Wait. The connection string to the database is sensitive data too, and we may need it again just like the administrator password. I suggest we add it to Key Vault as well.

**Mara:** Excellent idea! We can use the template to add the connection string to Key Vault. This means we need the name of the key vault in the parameter file too. The template will need it.

Andy updates the whiteboard.



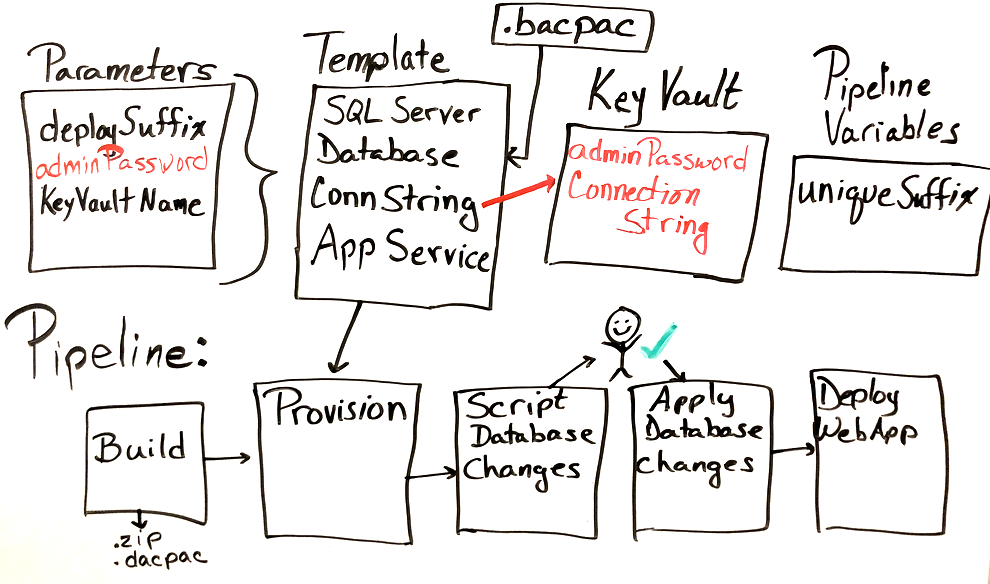
**Tim:** Speaking of the pipeline, what changes do we need to make there?

### Add a Resource Manager template to Azure Pipelines

Adding a Resource Manager template to the pipeline is going to require a few changes to the current setup. The team needs to consider the new tasks that it needs and how to work with Azure Key Vault. There's also a connection string to configure. Let's continue to listen in as they talk about pipeline changes.

**Mara:** Since we're only prototyping this for now, we're going to need unique names for the SQL Server and App Service instances. Let's add a variable to Azure Pipelines that specifies a unique suffix for the SQL Server name and App Service name. That way we can be sure that if we run the template again, we have the same suffix. This ensures we don't break the idempotency of the template deployment. And we can use different variables for different deployments.

**Andy:** Right. I'll add pipeline variables to the diagram.



**Mara:** Now we need a task that runs the template. I found this one, AzureResourceManagerTemplateDeployment@3 . It has parameters for the template file and the template parameter file.

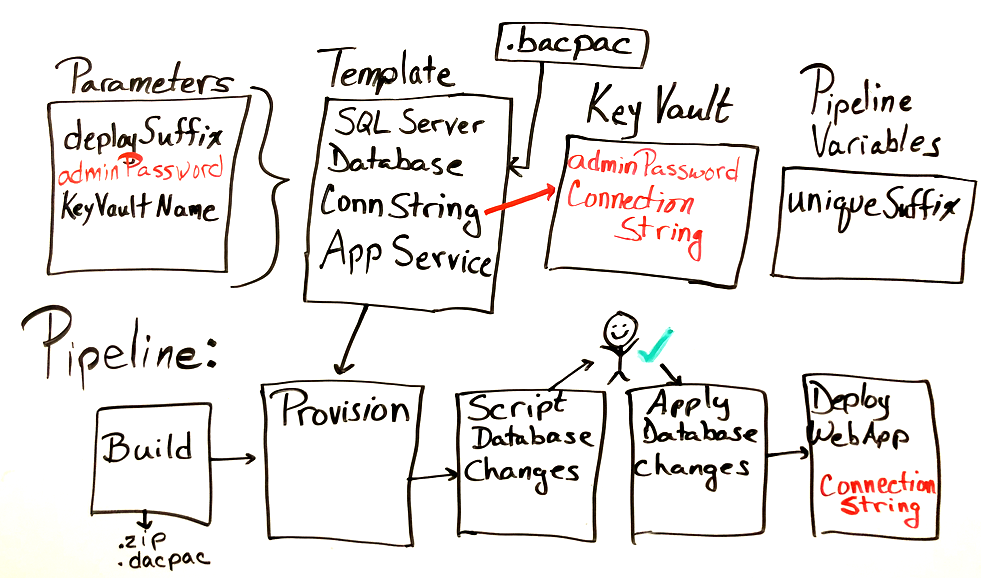
We also need to think about setting the connection string in the App Service instance. We could do that in the template, but then when we deploy the web app, it will get overwritten. So I think we need a task to set that after the web app is deployed.

**Andy:** I'm way ahead of you, Mara. I found the AzureAppServiceSettings@1  task. But it needs the connection string. How do I get that out of the key vault?

**Tim:** We can use the AzureKeyVault@1  task to fetch the connection string in the pipeline.

**Mara:** Excellent! This task will help us in a few stages.

Andy updates the whiteboard.



**Andy:** If there are no other concerns, then I think we have a plan.

**Tim:** Let's start with a basic template and run it from Cloud Shell in the Azure portal. We can pass in the parameters we need to create just the basic infrastructure. When we have that working, we can add the other pieces and deploy it from the pipeline. I'd like to lead on this part.

**Andy:** Great. Let me know how it turns out.