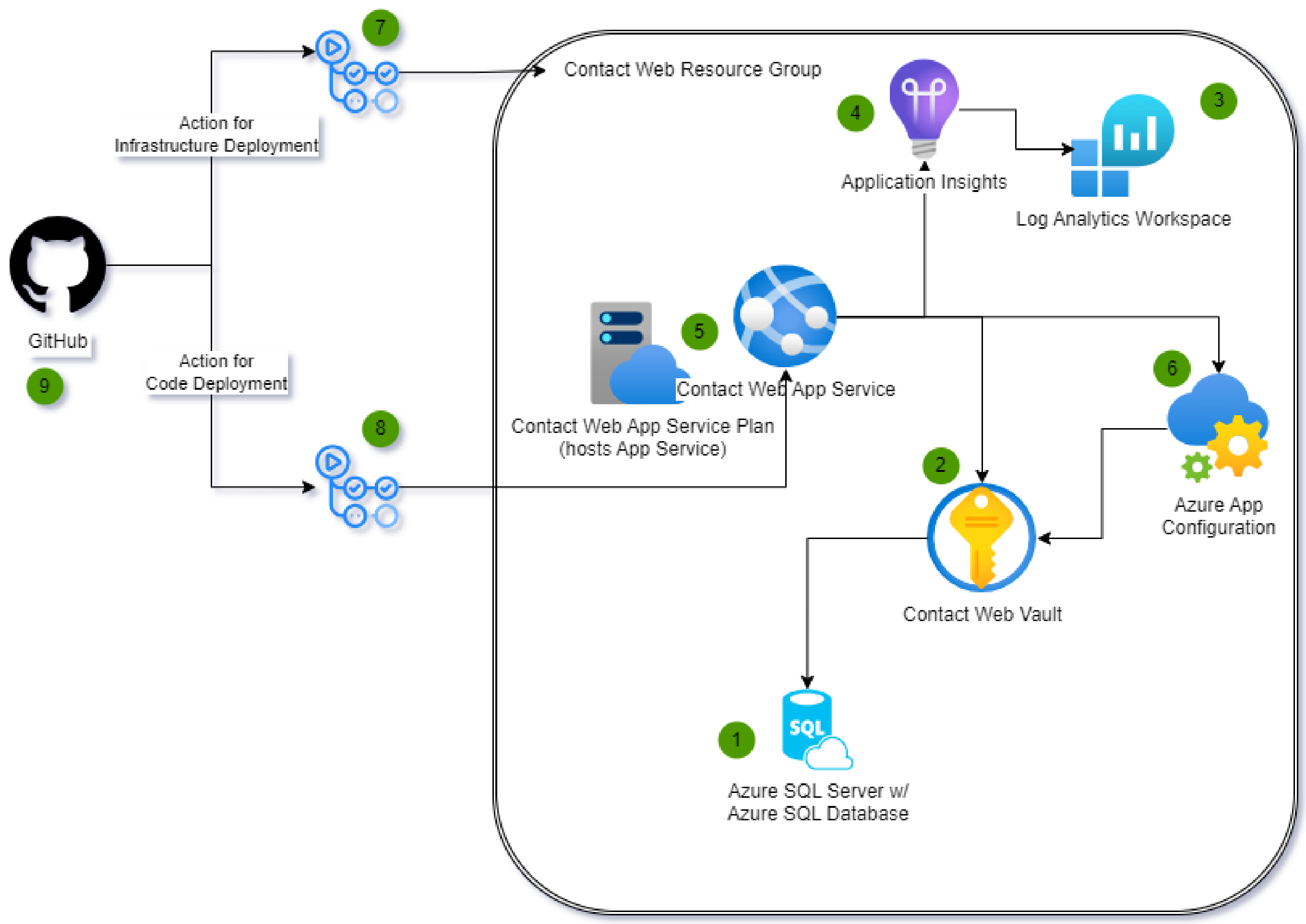
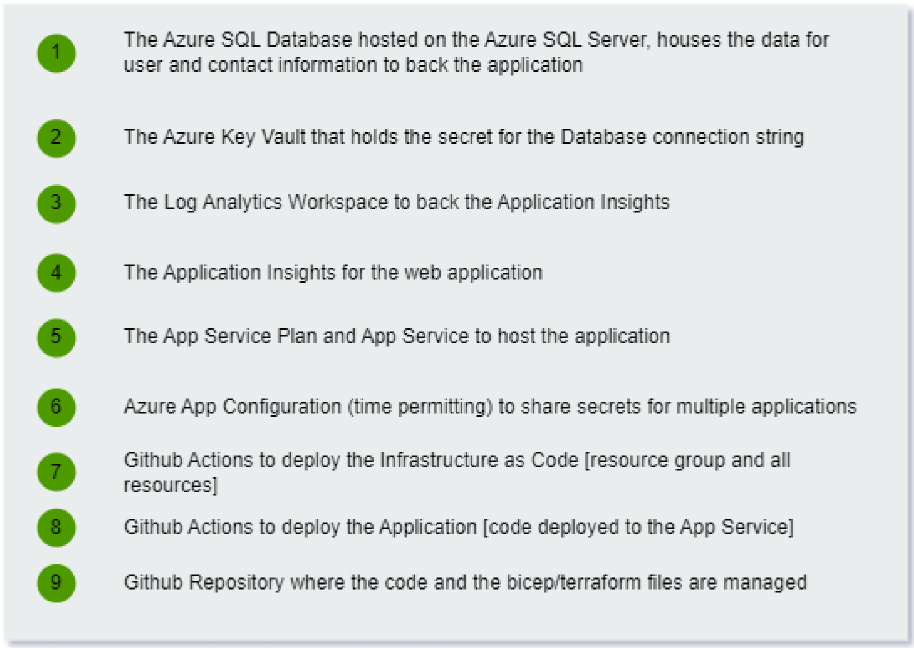
Part 2 - Building the Infrastructure

In this part you will build all the resources for implementation into Azure to host the application. You will use Bicep or Terraform to build the infrastructure.

As a reminder, this is the architecture we are building. By completing the `Part2-BuildingTheInfrastructure` walkthrough, you should already have the resource group for the application built.





1	The Azure SQL Database hosted on the Azure SQL Server, houses the data for user and contact information to back the application
2	The Azure Key Vault that holds the secret for the Database connection string
3	The Log Analytics Workspace to back the Application Insights
4	The Application Insights for the web application
5	The App Service Plan and App Service to host the application
6	Azure App Configuration (time permitting) to share secrets for multiple applications
7	Github Actions to deploy the Infrastructure as Code [resource group and all resources]
8	Github Actions to deploy the Application [code deployed to the App Service]
9	Github Repository where the code and the bicep/terraform files are managed

> *Note*: The Azure App Configuration will be added in Part 4 if there is enough time to complete the activities.

Prerequisites

You will need to have completed both Part 1 of this walkthrough and also the common Building The Infrastructure walkthrough before starting this walkthrough. If you do not already have a GitHub action that is deploying to Azure at the subscription level and creating a resource group, you will not be able to complete this walkthrough.

Other information

We will use the `deployContactWebArchitecture.tf` created in the common walkthrough as the main module for this deployment, all the other resources that we will deploy in the next steps will be added as modules. Lets see if you remember how we created modules in part 1!

Task 1 - Deploy Azure SQL Server

In this task you will get a your Azure SQL Server with a SQL database deployed. The SQL Server and the database will be a basic tier ($5/month) so even on the free account you shouldn't have to pay anything.

The database will be named `contactwebdb` and the server will be named `contactwebdbserverterraform`.

The server will be created with a firewall rule to allow all Azure services to access the server.

Step 1 - Create the Terraform module For the SQL Server

In this step you'll create the Terraform module for the SQL Server and database

1. Add a `sqlServer` folder inside the `modules` folder (create one if you don't have one already) and add the terraform files needed to deploy the database (main and variables).

2. Add the following text to the main module.

```
locals {
  sqlServerUniqueName   = "${var.sqlServerName}${var.uniqueIdentifier}"
}

# Azure SQL Server
resource "azurerm_mssql_server" cm_sql_server {
  name                                 = local.sqlServerUniqueName
  resource_group_name                  = var.resourceGroupName
  location                             = var.location
  version                              = "12.0"
  administrator_login                  = var.sqlServerAdmin
  administrator_login_password         = var.sqlServerPwd
  outbound_network_restriction_enabled  = false
  minimum_tls_version                  = "1.2"
  public_network_access_enabled        = true
}

resource "azurerm_mssql_firewall_rule" "allowAzureServices" {
  name             = "AllowAzureServices"
  server_id        = azurerm_mssql_server.cm_sql_server.id
  start_ip_address = "0.0.0.0"
  end_ip_address   = "0.0.0.0"
}

resource "azurerm_mssql_firewall_rule" "allowClientMachine" {
  name             = "AllowClientMachine"
  server_id        = azurerm_mssql_server.cm_sql_server.id
  start_ip_address = var.clientIpAddress
  end_ip_address   = var.clientIpAddress
}

resource "azurerm_mssql_database" "cm_db" {
  name                        = var.sqlDatabaseName
  server_id                   = azurerm_mssql_server.cm_sql_server.id
  collation                   = "SQL_Latin1_General_CP1_CI_AS"
  max_size_gb                 = 2
  read_scale                  = false
  sku_name                    = var.sqlDbSkuName
  zone_redundant              = false
  auto_pause_delay_in_minutes = 60
  ledger_enabled              = false
  min_capacity                = 0.5
}
```

> **Note**: As a reminder, all source files can be found in the `src/iac/terraform/Part2` folder from this repo in case something is not working as expected from this walkthrough.

1. Review the file to see what is happening

    - Creating the SQL Server
    - Adding firewall rules for Azure services and your client IP address
    - Creating the database
    - Parameters for the SQL Server and database name as well as the admin username and password

Step 2 - Create the variables file

In this step you'll create the variables file for the SQL Server and database

1. Create a `variables.tf` file in the `sqlServer` folder inside `modules`.

2. Add the following text to the parameters file:

```
variable "resourceGroupName" {
    type     = string
    nullable = false
}

variable "location" {
    type     = string
    nullable = false
}

variable "sqlServerName" {
    type     = string
    nullable = false
}
```

```
variable "sqlServerAdmin" {
    type     = string
    nullable = false
}

variable "sqlServerPwd" {
    type     = string
    nullable = false
}

variable "sqlDatabaseName" {
    type     = string
    nullable = false
}

variable "uniqueIdentifier" {
    type     = string
    nullable = false
}

variable "clientIpAddress" {
    type     = string
    nullable = false
}

variable "sqlDbSkuName" {
    type     = string
    nullable = false

    validation {
      condition     = contains(["Basic"], var.sqlDbSkuName)
      error_message = "Only SKU allowed for DB is 'Basic'"
    }
}
```

Note that we are adding a validation rule for the DB SKU name to avoid using any expensive tiers by accident.

1. Copy the parameters above into the `variables.tf` file for the main module and assign values using the `terraform.tfvars` file.

Step 3 - Create outputs file

1. Add an `outputs.tf` file to your module with the following content:

```
output "sqlServerName" {
  value = azurerm_mssql_server.cm_sql_server.name
}
```

Step 4 - Call SQL server module from main module.

Now that we have our SQL server module ready, we need to call it from the main module.

1. Add the following block to your `deployContactWebArchitecture.tf` file:

```
module "sqlServer" {
  source = "./modules/sqlServer"

  resourceGroupName = azurerm_resource_group.rg-contact-web-application.name
  location          = azurerm_resource_group.rg-contact-web-application.location
  clientIpAddress   = var.clientIpAddress
  sqlServerName     = var.sqlServerName
  sqlServerAdmin    = var.sqlServerAdmin
  sqlServerPwd      = var.sqlServerPwd
  sqlDatabaseName   = var.sqlDatabaseName
  sqlDbSkuName      = var.sqlDbSkuName
  uniqueIdentifier  = var.uniqueIdentifier
}
```

As you can see, we are using the resource group name and location from the existing resource group block.

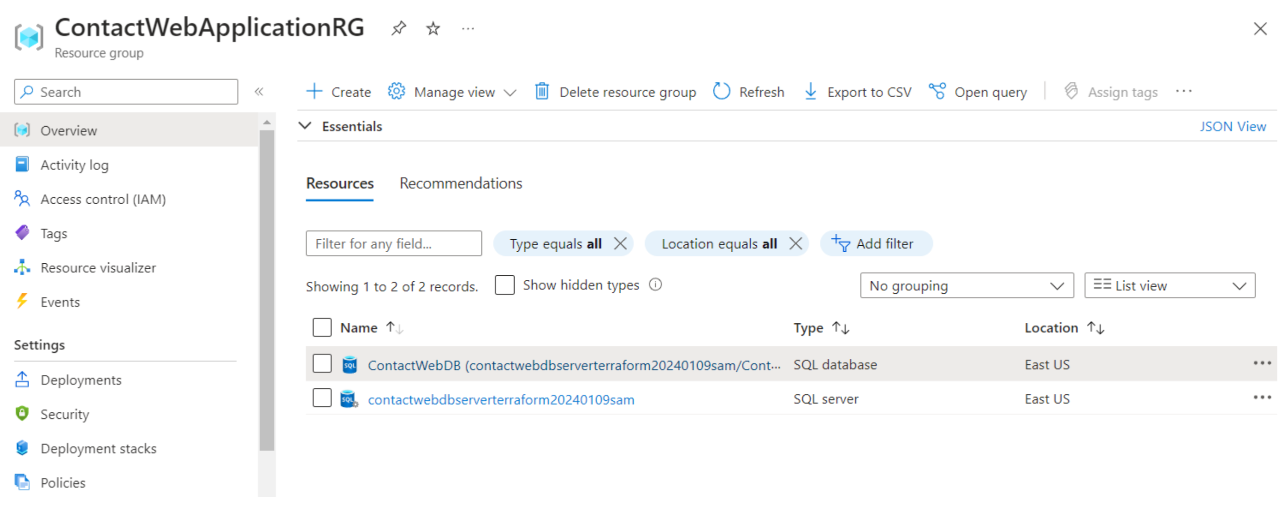
Step 4 - Check in and deploy

Check in all the files you just created and the deployment should run automatically.

1. Deploy by checking in and pushing your changes.

2. Validate Server

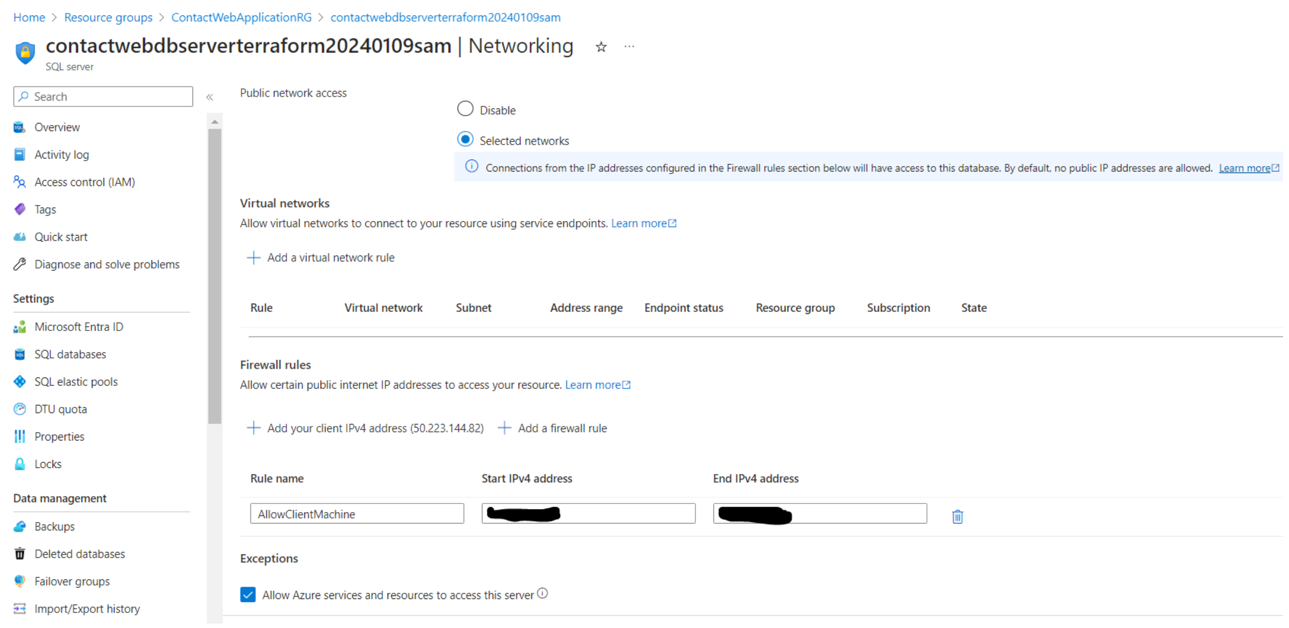
   Navigate to the resource group. You should see something like the following:



Showing 1 to 2 of 2 records.

Name	Type	Location
ContactWebDB (contactwebdbserverterraform20240109sam/Cont...	SQL database	East US
contactwebdbserverterraform20240109sam	SQL server	East US

   Open the `SQL Server` and click on `Networking`. You should see something like the following:



contactwebdbserverterraform20240109sam | Networking

Public network access

Disable
[✓ Selected networks]

Connections from the IP addresses configured in the Firewall rules section below will have access to this database. By default, no public IP addresses are allowed. Learn more

Virtual networks

Allow virtual networks to connect to your resource using service endpoints. Learn more

+ Add a virtual network rule

Rule	Virtual network	Subnet	Address range	Endpoint status	Resource group	Subscription	State

Firewall rules

Allow certain public internet IP addresses to access your resource. Learn more

+ Add your client IPv4 address (50.223.144.82) + Add a firewall rule

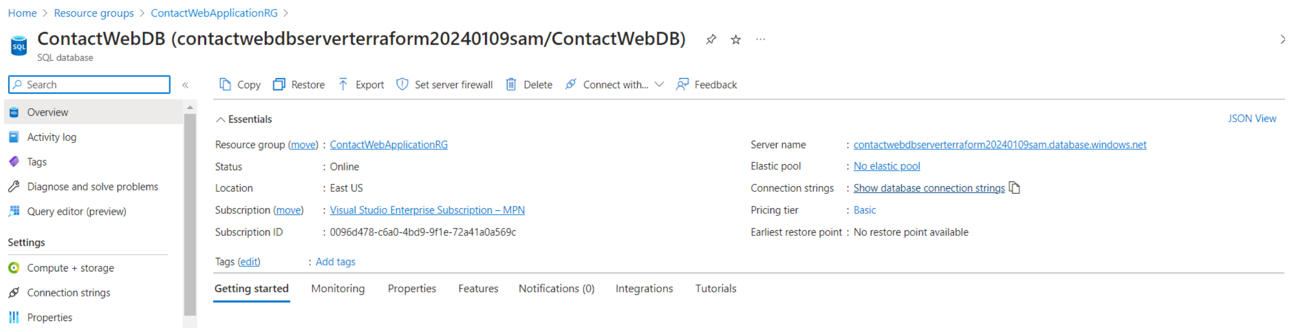
Rule name	Start IPv4 address	End IPv4 address
AllowClientMachine	[redacted]	[redacted]

Exceptions

[✓] Allow Azure services and resources to access this server

3. Validate Database

   Return to the request group and click on the `SQL Database`, or go to the database from the server.



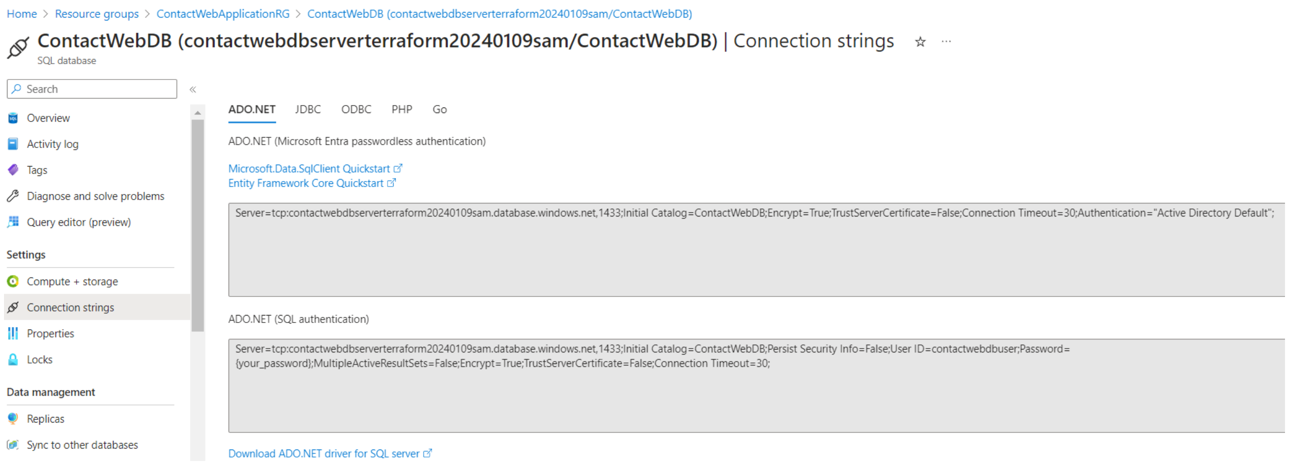
ContactWebDB (contactwebdbserverterraform20240109sam/ContactWebDB)

Essentials

Resource group (move)	: ContactWebApplicationRG	Server name	: contactwebdbserverterraform20240109sam.database.windows.net
Status	: Online	Elastic pool	: No elastic pool
Location	: East US	Connection strings	: Show database connection strings
Subscription (move)	: Visual Studio Enterprise Subscription – MPN	Pricing tier	: Basic
Subscription ID	: 0096d478-c6a0-4bd9-9f1e-72a41a0a569c	Earliest restore point	: No restore point available
Tags (edit)	: Add tags		

4. Note Connection string

   On the database connection string info, validate that you have a database with the name you specified and that you can see the connection string with all the info except the password:


Home > Resource groups > ContactWebApplicationRG > ContactWebDB (contactwebdbserverterraform20240109sam/ContactWebDB)
SQL database
Search
Overview
Activity log
Tags
Diagnose and solve problems
Query editor (preview)
Settings
Compute + storage
Connection strings
Properties
Locks
Data management
Replicas
Sync to other databases
ADO.NET
JDBC
ODBC
PHP
Go
Microsoft.Data.SqlClient Quickstart
Entity Framework Core Quickstart
Download ADO.NET driver for SQL server

ContactWebDB (contactwebdbserverterraform20240109sam/ContactWebDB) | Connection strings

ADO.NET (Microsoft Entra passwordless authentication)

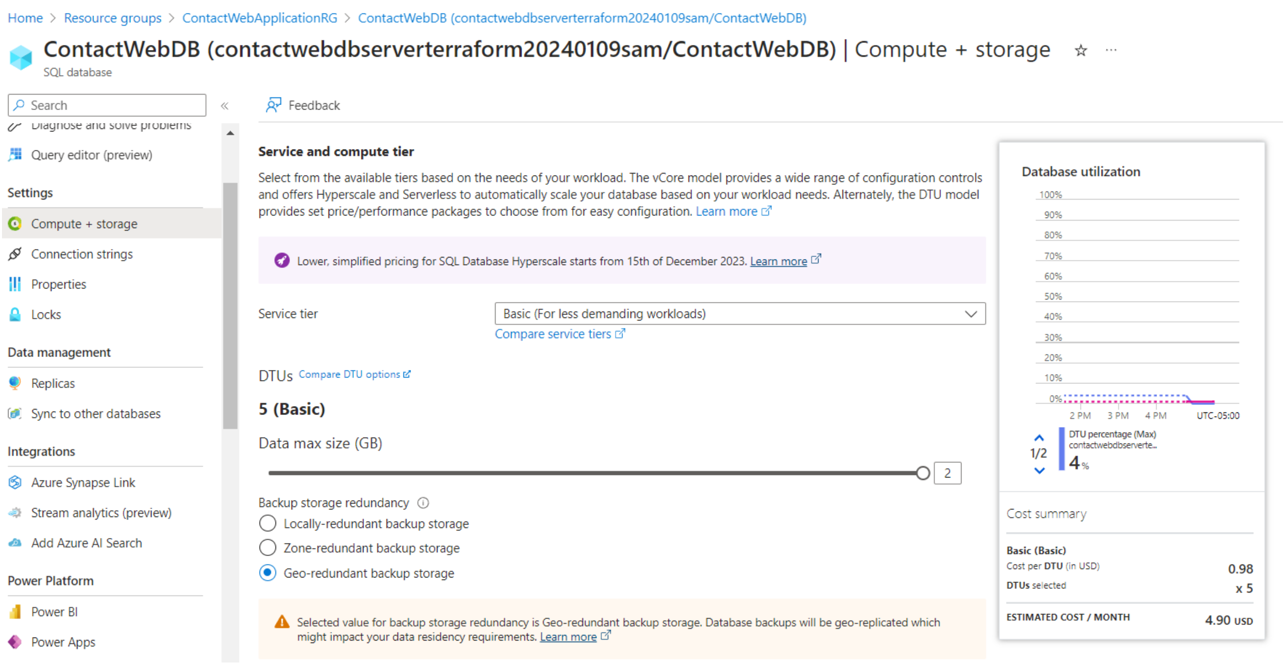
```
Server=tcp:contactwebdbserverterraform20240109sam.database.windows.net,1433;Initial Catalog=ContactWebDB;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;Authentication="Active Directory Default";
```

ADO.NET (SQL authentication)

```
Server=tcp:contactwebdbserverterraform20240109sam.database.windows.net,1433;Initial Catalog=ContactWebDB;Persist Security Info=False;User ID=contactwebdbuser;Password=
{your_password};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;
```

5. Validate Database SKU

   Make sure that you didn't accidentally create a database with a higher SKU than you wanted. You should see something like the following:


Home > Resource groups > ContactWebApplicationRG > ContactWebDB (contactwebdbserverterraform20240109sam/ContactWebDB)

ContactWebDB (contactwebdbserverterraform20240109sam/ContactWebDB) | Compute + storage

SQL database

Service and compute tier

Select from the available tiers based on the needs of your workload. The vCore model provides a wide range of configuration controls and offers Hyperscale and Serverless to automatically scale your database based on your workload needs. Alternately, the DTU model provides set price/performance packages to choose from for easy configuration. Learn more

Lower, simplified pricing for SQL Database Hyperscale starts from 15th of December 2023. Learn more

Service tier: Basic (For less demanding workloads)

Compare service tiers

DTUs Compare DTU options

5 (Basic)

Data max size (GB): 2

Backup storage redundancy
- ( ) Locally-redundant backup storage
- ( ) Zone-redundant backup storage
- (●) Geo-redundant backup storage

Selected value for backup storage redundancy is Geo-redundant backup storage. Database backups will be geo-replicated which might impact your data residency requirements. Learn more

Database utilization

Cost summary

Basic (Basic)	
Cost per DTU (in USD)	0.98
DTUs selected	x 5
ESTIMATED COST / MONTH	4.90 USD

> **Note:** we will use IAC to get and set the connection string information into the Key Vault in a future step, so you don't need to copy it or save it anywhere.

Task 2 - Log analytics/Application Insights

In order to log analytics for the web application, we need to create application insights backed by log analytics workspace. We will create the log analytics workspace first and then create the application insights, then finally create the app service.

Step 1 - Create Log Analytics Workspace

1. Add a new module (folder and files) for the analytics workspace, you will need the following files:

- Main file (logAnalytics.tf)
- Variables file.
- Outputs file.

1. Add the following text to the `logAnalytics.tf` file.

```
resource "azurerm_log_analytics_workspace" "cm_analytics_workspace" {
  name                              = var.logAnalyticsWorkSpaceName
  location                          = var.location
  resource_group_name               = var.resourceGroupName
  sku                               = "PerGB2018"
  retention_in_days                 = 30
  allow_resource_only_permissions = true
}
```

Step 2 - Update variables file

1. Add the resource group name, location and log analytics name as variables for the module.

Step 3 - Add outputs to file

1. Add the following to the outputs.tf file to output the workspace Id:

```
output "logAnalyticsWorkspaceId" {
  value = azurerm_log_analytics_workspace.cm_analytics_workspace.id
}
```

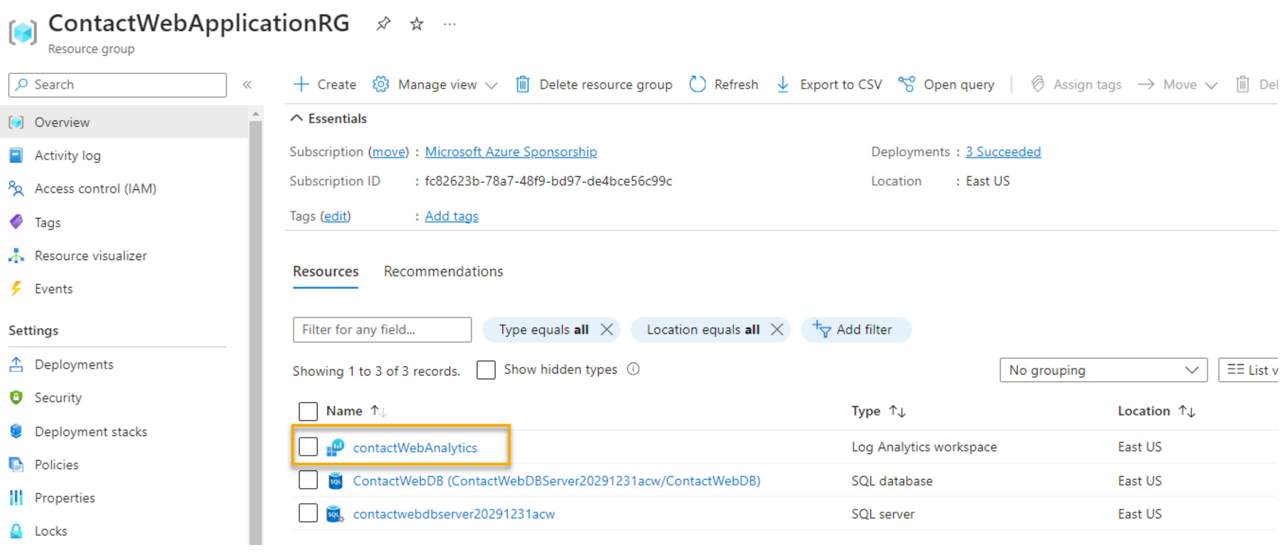
Step 4 - Deploy Log Analytics for the Web Application's Application Insights

Deploy the log analytics workspace.

1. Add the analytics module to the root module in your deployment (use the SQL server block as an example).

2. Add the analytics workspace name to the `variables.tf` of the root module

3. Check in and let the deployment run to validate your log analytics workspace (you can start creating the next one while it builds).



Subscription (move) : Microsoft Azure Sponsorship
Subscription ID : fc82623b-78a7-48f9-bd97-de4bce56c99c
Tags (edit) : Add tags
Deployments : 3 Succeeded
Location : East US

Showing 1 to 3 of 3 records.

Name	Type	Location
contactWebAnalytics	Log Analytics workspace	East US
ContactWebDB (ContactWebDBServer20291231acw/ContactWebDB)	SQL database	East US
contactwebdbserver20291231acw	SQL server	East US

Step 4 - Create Application Insights

In this step you'll build the application insights resource that will be backed by the log analytics workspace built in the previous step.

1. Create a new module for the application insights with the following file:

- Main file (applicationInsights.tf).
- Variables file.
- Outputs file.

1. Add the following text to the main file.

```
```

```
resource "azurerm_application_insights" "cm_app_insights" {
  name                = var.appInsightsName
  location            = var.location
  resource_group_name = var.resourceGroupName
  workspace_id        = var.logAnalyticsWorkspaceId
  application_type    = "web"
}
```

Step 5 - Create the parameters file

Create the parameters file.

1. Add the resource group name, location, application insights resource name and workspaceId as variables for the module.

Step 6: Deploy

Deploy the application insights.

1. Add the deployment to the main deployment file

   Add the following to the main deployment file:

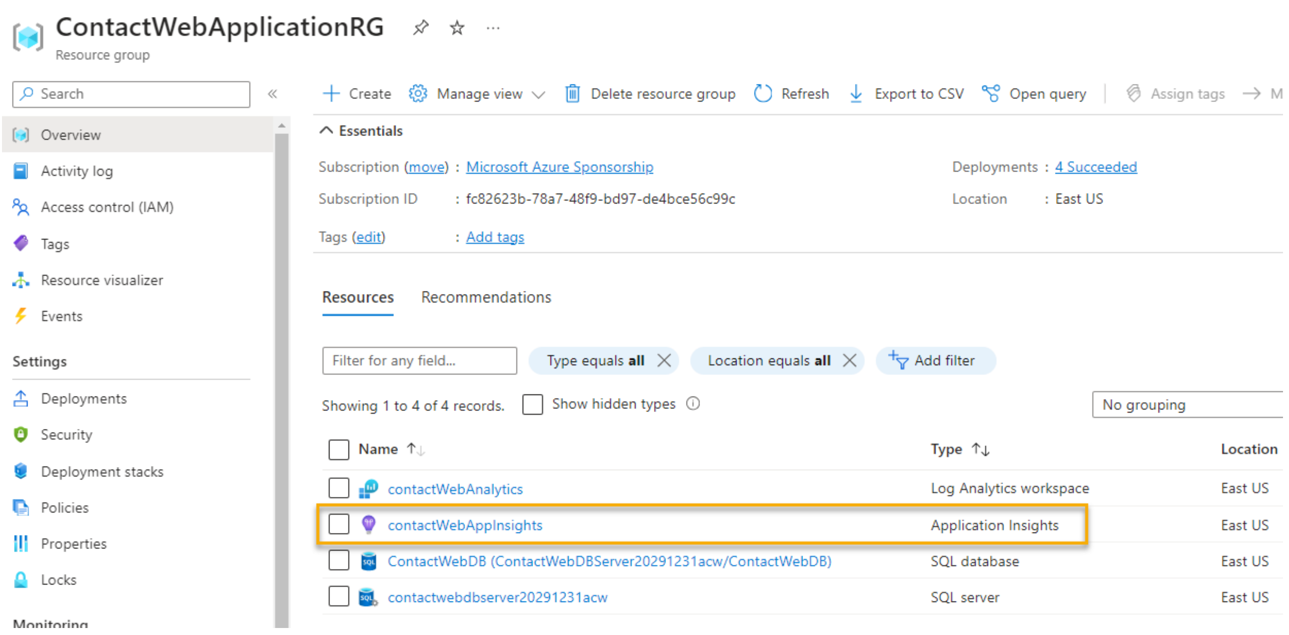
```
module "applicationInsights" {
  source = "./modules/appInsights"

  resourceGroupName          = azurerm_resource_group.rg-contact-web-
application.name
  location                   = azurerm_resource_group.rg-contact-web-
application.location
  logAnalyticsWorkspaceId    = module.logAnalyticsWorkspace.logAnalyticsWorkspaceId
  appInsightsName            = var.appInsightsName
}
```

> **Note:** Notice the dependency on the log analytics workspace. This is how you can ensure that the log analytics workspace is created before the application insights.

Don't forget to add the new variable for the application insights resource to the main module!

1. Check in and deploy changes

   Navigate back to the resource group and review the resources.



ContactWebApplicationRG
Resource group

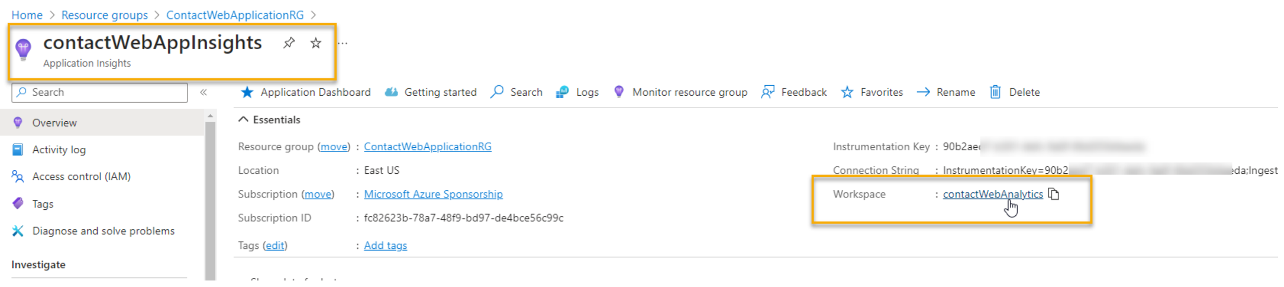
Subscription (move) : Microsoft Azure Sponsorship
Subscription ID : fc82623b-78a7-48f9-bd97-de4bce56c99c
Tags (edit) : Add tags
Deployments : 4 Succeeded
Location : East US

Showing 1 to 4 of 4 records.

Name	Type	Location
contactWebAnalytics	Log Analytics workspace	East US
contactWebAppInsights	Application Insights	East US
ContactWebDB (ContactWebDBServer20291231acw/ContactWebDB)	SQL database	East US
contactwebdbserver20291231acw	SQL server	East US

2. Validate that the Application Insights is correctly leveraging the workspace

   Open the Application Insights and review the `overview` blade.



Home > Resource groups > ContactWebApplicationRG >
contactWebAppInsights
Application Insights

Resource group (move) : ContactWebApplicationRG
Location : East US
Subscription (move) : Microsoft Azure Sponsorship
Subscription ID : fc82623b-78a7-48f9-bd97-de4bce56c99c
Tags (edit) : Add tags
Instrumentation Key : 90b2ae[redacted]
Connection String : InstrumentationKey=90b2[redacted]da;Ingest
Workspace : contactWebAnalytics

Task 3 - Key vault

In this task you will create the key vault and add the secrets for the database connection strings, if you already went through the Bicep walkthrough you'll notice that we are doing it before the web app to show that there are different ways to deploy your resources. We will also use RBAC instead of policies to give access to the vault.

Step 1 - Create the Key Vault

Create the Key Vault. You need to get the following accomplished with the deployment:

- Get the connection string from the database as a secret
- Create the vault with the two secrets and identity permission for the web app
- Give restricted key vault access to the identity running the deployment.

1. Create a new module for the key vault:

- Main file (keyvault.tf).
- Variables file.
- Outputs file.

1. Add the following to the file:

```
data "azurerm_client_config" "current" {}

data "azurerm_mssql_server" "cm_sql_server" {
  resource_group_name = var.resourceGroupName
  name                = "${var.sqlServerName}${var.uniqueIdentifier}"
}

locals {
  dbConnectionString =
"Server=tcp:${data.azurerm_mssql_server.cm_sql_server.fully_qualified_domain_name}
,1433;Initial Catalog=${var.sqlDatabaseName};Persist Security Info=False;User
ID=${data.azurerm_mssql_server.cm_sql_server.administrator_login};Password=${var.s
qlServerPwd};MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=Fa
lse;Connection Timeout=30;"
}

resource "azurerm_key_vault" "cm_kv" {
  name                            = "${var.keyVaultName}${var.uniqueIdentifier}"
  location                        = var.location
  resource_group_name             = var.resourceGroupName
  tenant_id                       = data.azurerm_client_config.current.tenant_id
  sku_name                        = "standard"
  soft_delete_retention_days      = 7
  purge_protection_enabled        = true
  enable_rbac_authorization       = true
  enabled_for_template_deployment = true
}

resource "azurerm_key_vault_secret" "kv_secret_identity" {
  name         = "IdentityDbConnectionSecret"
  value        = local.dbConnectionString
  key_vault_id = azurerm_key_vault.cm_kv.id

  depends_on = [ azurerm_role_assignment.kv_access_current_user ]
}

resource "azurerm_key_vault_secret" "kv_secret_cm" {
  name         = "ContactManagerDbConnectionSecret"
  value        = local.dbConnectionString
  key_vault_id = azurerm_key_vault.cm_kv.id

  depends_on = [ azurerm_role_assignment.kv_access_current_user ]
}

resource "azurerm_role_assignment" "kv_access_current_user" {
  scope                = azurerm_key_vault.cm_kv.id
  role_definition_name = "Key Vault Secrets Officer"
  principal_id         = data.azurerm_client_config.current.object_id
}
```

There is one new element in this file that we haven't used in any of the previous resources: depends_on, this attribute allows you to specify certain dependencies between resources to ensure that they are deployed in the right order. In this case, we are saying that we want to try to create the secrets only after the appropriate permissions have been given to the current user.

Step 2 - Add variables

1. Add the following text to the variables file:

```
variable "resourceGroupName" {
    type     = string
    nullable = false
}

variable "location" {
    type     = string
    nullable = false
}

variable "sqlServerName" {
    type     = string
    nullable = false
}

variable "sqlServerPwd" {
    type      = string
    nullable  = false
    sensitive = true
}

variable "sqlDatabaseName" {
    type     = string
    nullable = false
}

variable "keyVaultName" {
    type      = string
    nullable  = false
}

variable "uniqueIdentifier" {
    type      = string
    nullable  = false
}
```

Step 3 - Deploy

Add a reference to the keyvault module in the root module.

1. Add the deployment to the root module

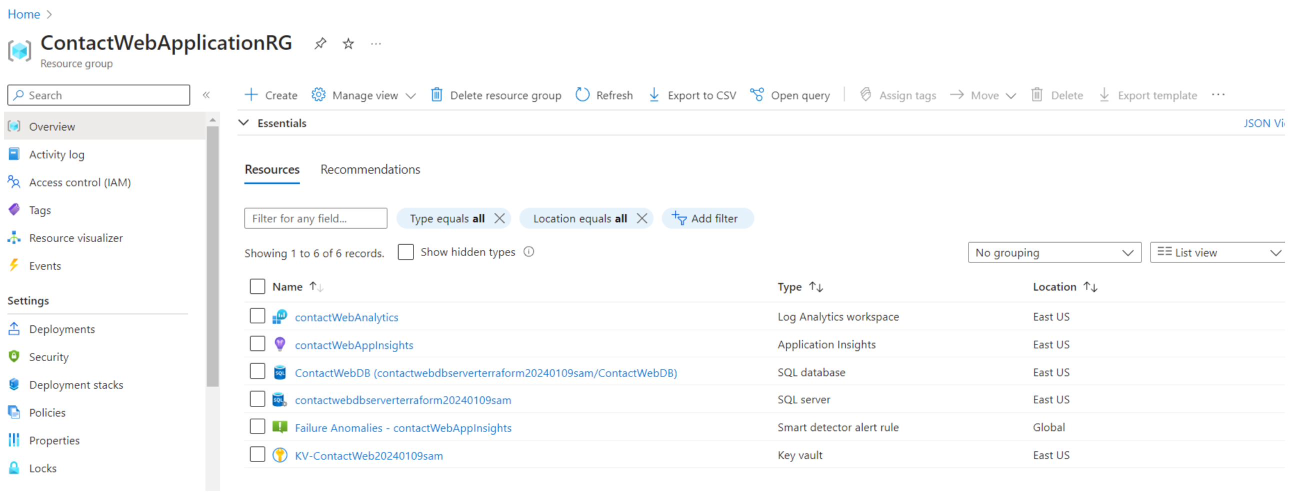
Add the following to the root module:

```
module "keyvault" {
  source = "./modules/keyvault"

  resourceGroupName = azurerm_resource_group.rg-contact-web-application.name
  location          = azurerm_resource_group.rg-contact-web-application.location
  sqlServerName     = var.sqlServerName
  sqlServerPwd      = var.sqlServerPwd
  sqlDatabaseName   = var.sqlDatabaseName
  uniqueIdentifier  = var.uniqueIdentifier
  keyVaultName      = var.keyVaultName
}
```

Don't forget to add the new variable (keyVaultName) to the `variables.tf` file in the root module and assign a value in the `terraform.tfvars`

1. Validate the deployment of the vault

   The vault should deploy as required.

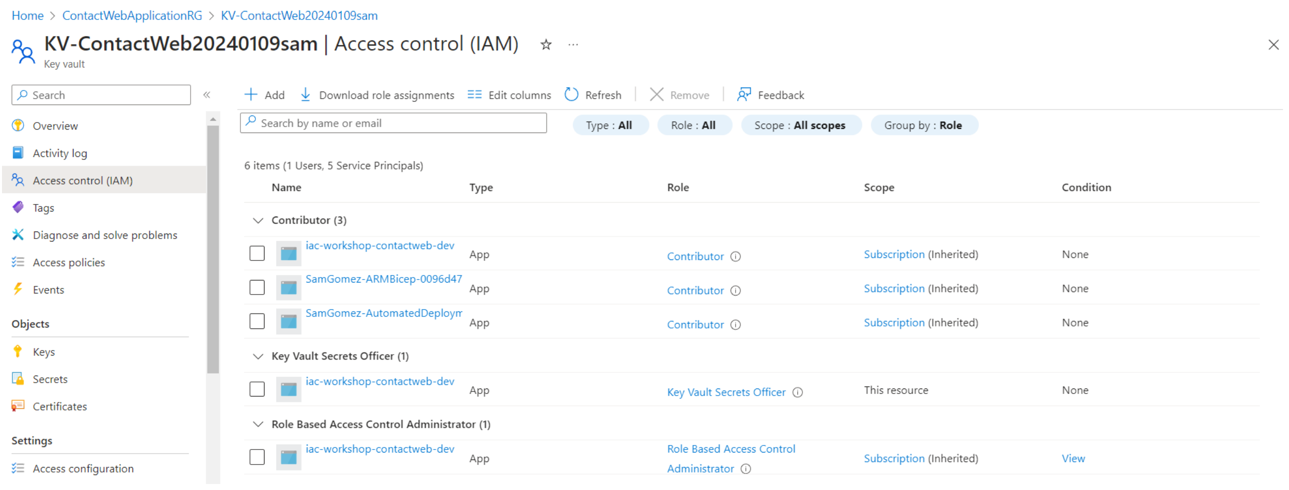


Showing 1 to 6 of 6 records. Show hidden types

Name	Type	Location
contactWebAnalytics	Log Analytics workspace	East US
contactWebAppInsights	Application Insights	East US
ContactWebDB (contactwebdbserverterraform20240109sam/ContactWebDB)	SQL database	East US
contactwebdbserverterraform20240109sam	SQL server	East US
Failure Anomalies - contactWebAppInsights	Smart detector alert rule	Global
KV-ContactWeb20240109sam	Key vault	East US

2. Verify the role assignments (make sure web app is authorized)

   Validate that the Web App has access to get secrets from the Key Vault

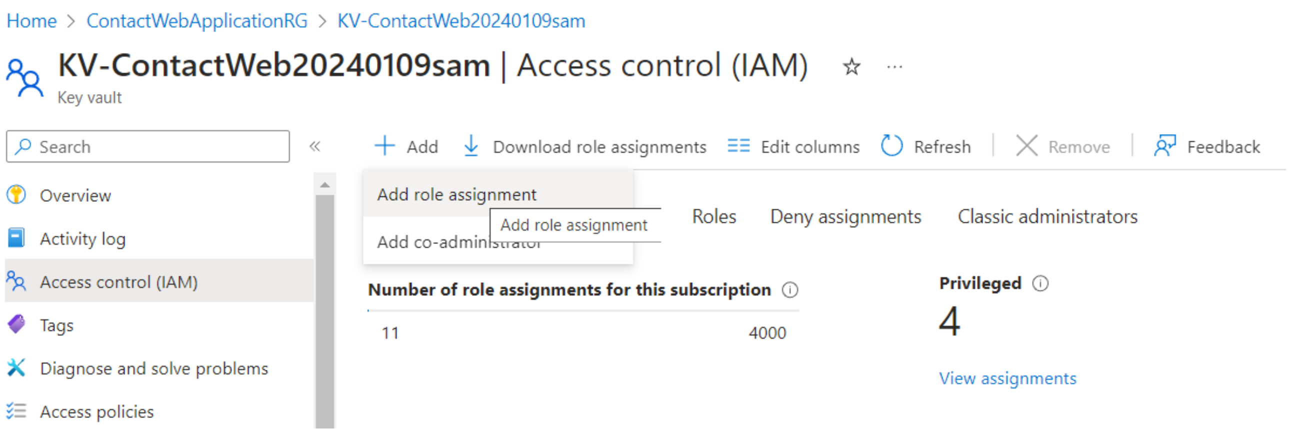


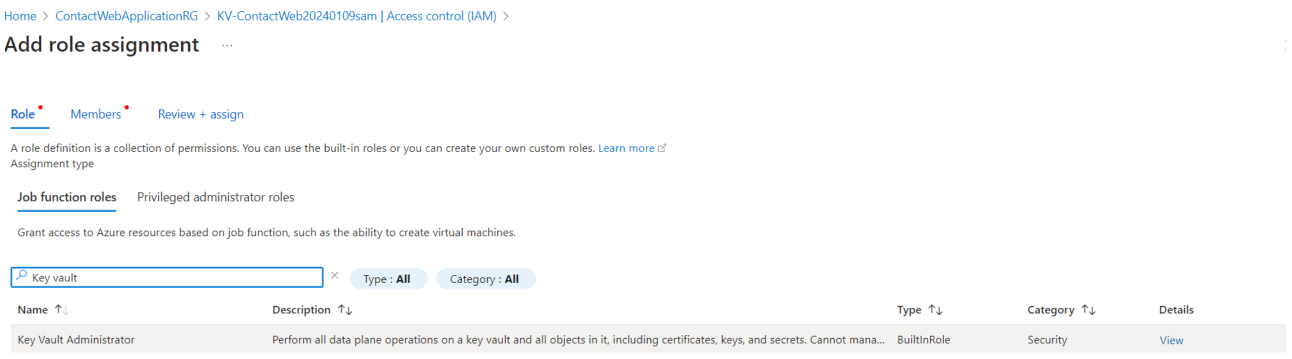
6 items (1 Users, 5 Service Principals)

Name	Type	Role	Scope	Condition
Contributor (3)				
iac-workshop-contactweb-dev	App	Contributor	Subscription (Inherited)	None
SamGomez-ARMBicep-0096d47	App	Contributor	Subscription (Inherited)	None
SamGomez-AutomatedDeploym	App	Contributor	Subscription (Inherited)	None
Key Vault Secrets Officer (1)				
iac-workshop-contactweb-dev	App	Key Vault Secrets Officer	This resource	None
Role Based Access Control Administrator (1)				
iac-workshop-contactweb-dev	App	Role Based Access Control Administrator	Subscription (Inherited)	View

3. Verify secrets (need to add your credential)

   Modify the role assignments to allow your user to view the secrets. You can do this by adding giving your user the `Key Vault Administrator` or `Key Vault Secrets Officer` role. Then you can view the secrets.

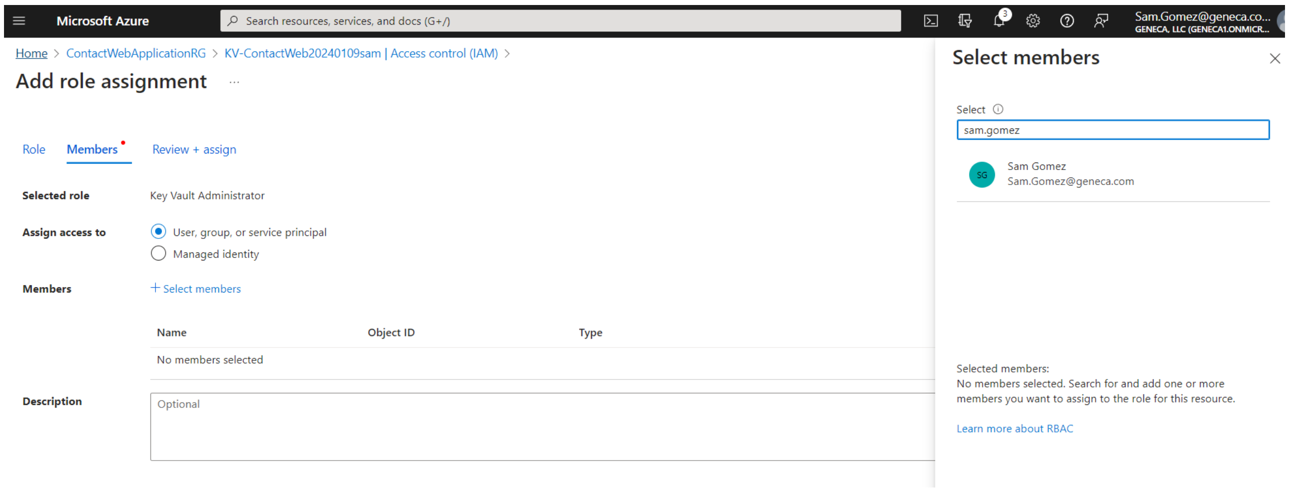



Home > ContactWebApplicationRG > KV-ContactWeb20240109sam | Access control (IAM) >
Add role assignment
Role
Members
Review + assign
Job function roles
Privileged administrator roles
Key vault
Type : All
Category : All

A role definition is a collection of permissions. You can use the built-in roles or you can create your own custom roles. Learn more
Assignment type

Grant access to Azure resources based on job function, such as the ability to create virtual machines.

Name	Description	Type	Category	Details
Key Vault Administrator	Perform all data plane operations on a key vault and all objects in it, including certificates, keys, and secrets. Cannot mana...	BuiltInRole	Security	View



Microsoft Azure | Search resources, services, and docs (G+/) | Sam.Gomez@geneca.co... GENECA, LLC (GENECA1.ONMICR...

Home > ContactWebApplicationRG > KV-ContactWeb20240109sam | Access control (IAM) >

Add role assignment ...

Role | Members | Review + assign

Selected role	Key Vault Administrator
Assign access to	[✓ User, group, or service principal] Managed identity
Members	+ Select members

Name	Object ID	Type
No members selected		

Description: Optional

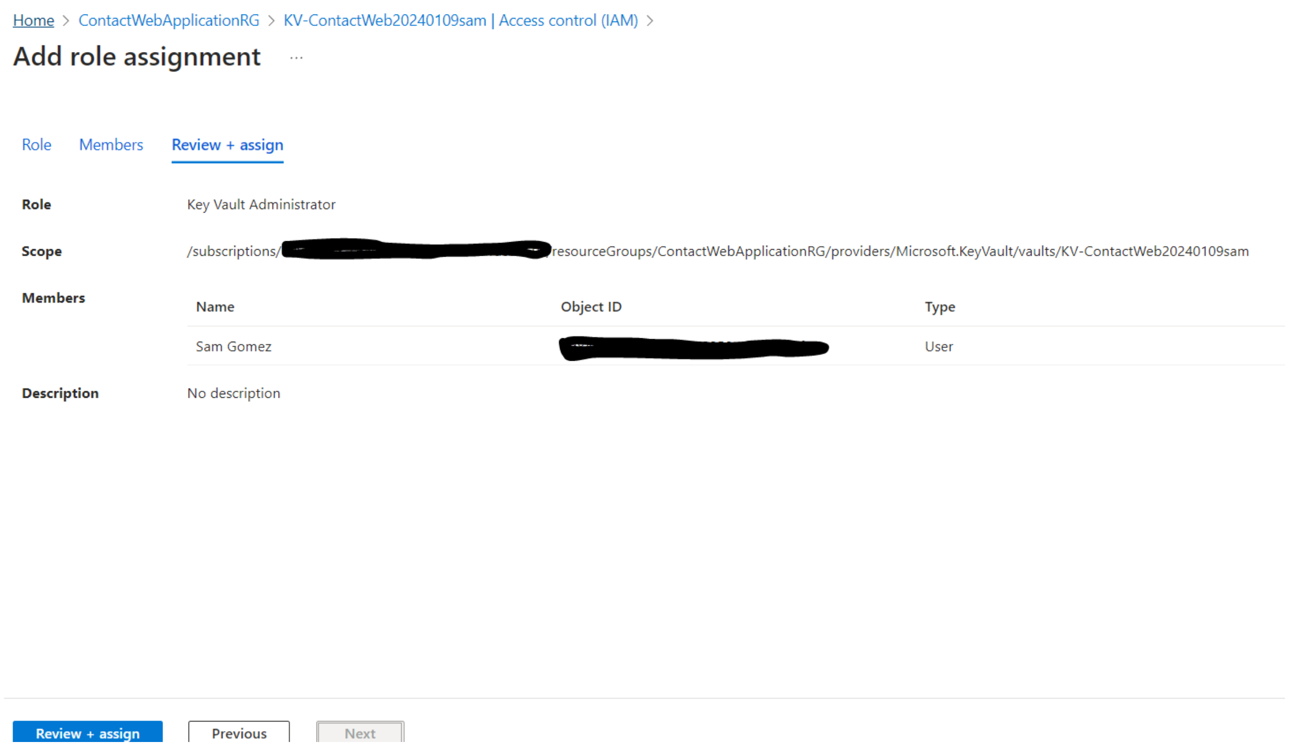
Select members

Select: sam.gomez

Sam Gomez
Sam.Gomez@geneca.com

Selected members:
No members selected. Search for and add one or more members you want to assign to the role for this resource.

Learn more about RBAC



Home > ContactWebApplicationRG > KV-ContactWeb20240109sam | Access control (IAM) >

Add role assignment ...

Role | Members | Review + assign

Role	Key Vault Administrator
Scope	/subscriptions/[redacted]resourceGroups/ContactWebApplicationRG/providers/Microsoft.KeyVault/vaults/KV-ContactWeb20240109sam

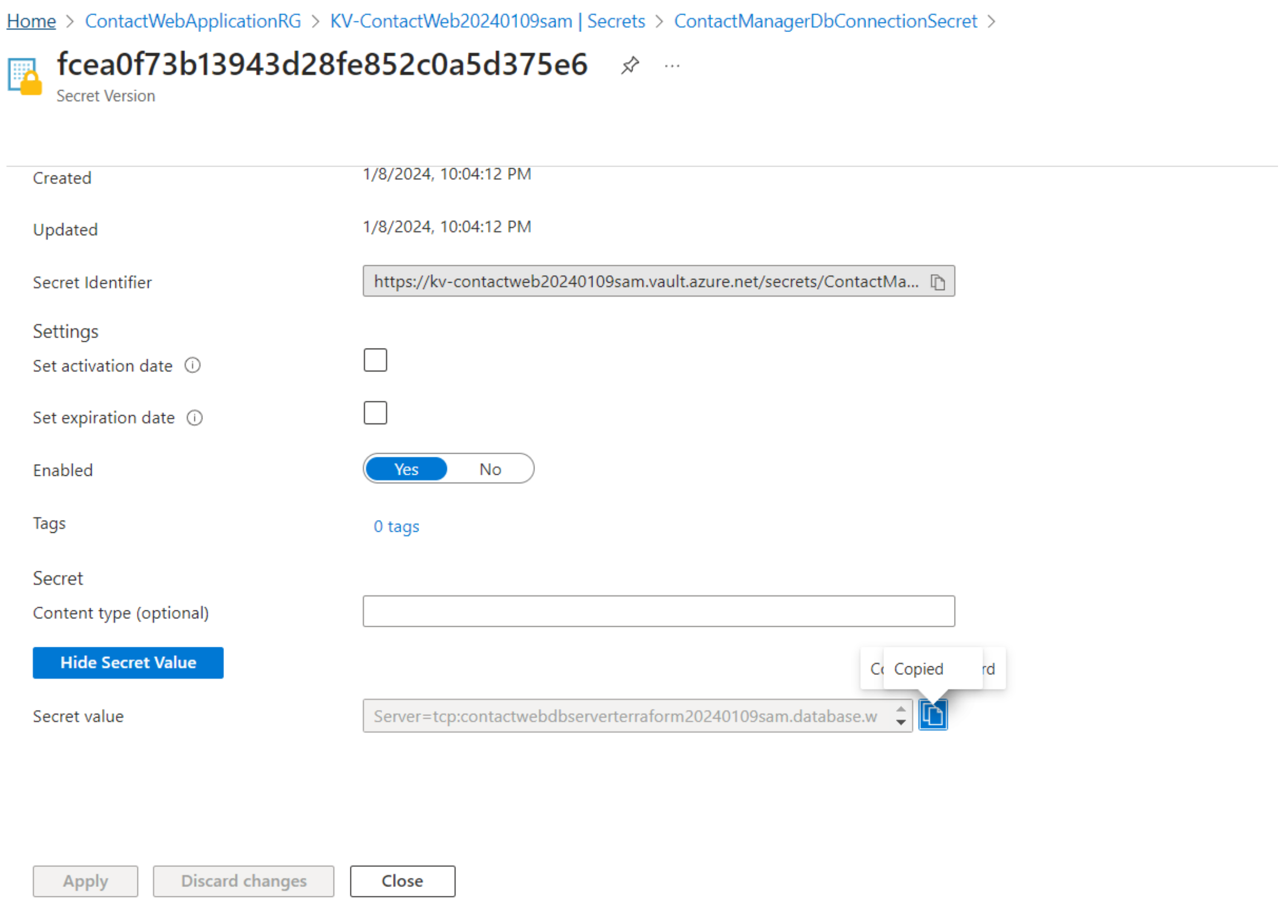
Members

Name	Object ID	Type
Sam Gomez	[redacted]	User

Description: No description

Review + assign | Previous | Next

Open the secrets and validate they have the correct value as expected.


Home > ContactWebApplicationRG > KV-ContactWeb20240109sam | Secrets > ContactManagerDbConnectionSecret >

fcea0f73b13943d28fe852c0a5d375e6

Secret Version

Created	1/8/2024, 10:04:12 PM
Updated	1/8/2024, 10:04:12 PM
Secret Identifier	https://kv-contactweb20240109sam.vault.azure.net/secrets/ContactMa...
Settings	
Set activation date	☐
Set expiration date	☐
Enabled	[✓ Yes] No
Tags	0 tags
Secret	
Content type (optional)	
Secret value	Server=tcp:contactwebdbserverterraform20240109sam.database.w

Task 4 - Create App Service Plan and App service

At this point, the backing database and log analytics/application insights are in place. We will need to leverage them in the App Service for the two database connections and the log analytics connection string/instrumentation key.

Step 1 - Create the App Service Plan and Site

1. Review necessary information

   In order for this to work, we need to set the following values in the configuration for the app service:

   - `ConnectionStrings:DefaultConnection` - this will be the connection string for the identity database
   - `ConnectionStrings:MyContactManager` - this will be the connection string for the database
   - `APPINSIGHTS:CONNECTIONSTRING` - this will be the instrumentation key for the application insights

   These database values will be leveraged a couple of other times and it's important not to make a *typo* on these so they will need to be parameterized.

2. Create a new module for the app service and app service plan.

- Main file (appService.tf).
- Variables file.
- Output file.

1. Add the following text to the `appService.tf` file.

```
data "azurerm_application_insights" "cm_app_insights" {
  name                = var.appInsightsName
  resource_group_name = var.resourceGroupName
}

resource "azurerm_service_plan" "cm_hosting_plan" {
  name                = "${var.appServicePlanName}-${var.uniqueIdentifier}"
  resource_group_name = var.resourceGroupName
  location            = var.location
  os_type             = "Windows"
  sku_name            = "F1"
}

resource "azurerm_windows_web_app" "cm_webapp" {
  name                = "${var.webAppName}-${var.uniqueIdentifier}"
  resource_group_name = var.resourceGroupName
  location            = azurerm_service_plan.cm_hosting_plan.location
  service_plan_id     = azurerm_service_plan.cm_hosting_plan.id
  identity {
    type = "SystemAssigned"
  }

  site_config {
    always_on = false
    application_stack {
        dotnet_version = "v6.0"
        current_stack  = "dotnet"
    }
  }

  app_settings = {
    "ConnectionStrings:DefaultConnection" =
"@Microsoft.KeyVault(SecretUri=${var.defaultDBSecretURI})"
    "ConnectionStrings:MyContactManager"  =
"@Microsoft.KeyVault(SecretUri=${var.managerDBSecretURI})"
    "APPINSIGHTS:CONNECTIONSTRING"        =
data.azurerm_application_insights.cm_app_insights.connection_string
    "WEBSITE_ENABLE_SYNC_UPDATE_SITE"     = "true"
    "WEBSITE_RUN_FROM_PACKAGE"            = "1"
  }
}

resource "azurerm_role_assignment" "kv_access_webapp" {
  scope                = var.keyVaultId
  role_definition_name = "Key Vault Secrets User"
  principal_id         =
azurerm_windows_web_app.cm_webapp.identity[0].principal_id
}
```

Notice that apart from creating the web app service we are giving the resource permission to access the keyvault, we are limiting the permissions to just this particular key vault but you have the option to set those permissions at the resource group level as well.

Step 2 - Add variables to file.

1. Add the following text to the variables file.

```
variable "resourceGroupName" {
    type     = string
    nullable = false
}

variable "location" {
    type     = string
    nullable = false
}

variable "appInsightsName" {
    type     = string
    nullable = false
}

variable "uniqueIdentifier" {
    type     = string
    nullable = false
}

variable "appServicePlanName" {
    type     = string
    nullable = false
}

variable "webAppName" {
    type     = string
    nullable = false
}

variable "defaultDBSecretURI" {
    type     = string
    nullable = false
}

variable "managerDBSecretURI" {
    type     = string
    nullable = false
}

variable "keyVaultId" {
    type     = string
    nullable = false
}
```

Step 3 - Deploy

Add the variables and app service module to the root module

1. Add the deployment to the root module

   Add the following to the root module:

```
module "appService" {
  source = "./modules/appService"

  resourceGroupName  = azurerm_resource_group.rg-contact-web-application.name
  location           = azurerm_resource_group.rg-contact-web-application.location
  appInsightsName    = var.appInsightsName
  uniqueIdentifier   = var.uniqueIdentifier
  appServicePlanName = var.appServicePlanName
  webAppName         = var.webAppName
  defaultDBSecretURI = module.keyvault.identityDBConnectionSecretURI
  managerDBSecretURI = module.keyvault.managerDBConnectionSecretURI
  keyVaultId         = module.keyvault.keyVaultId
}
```

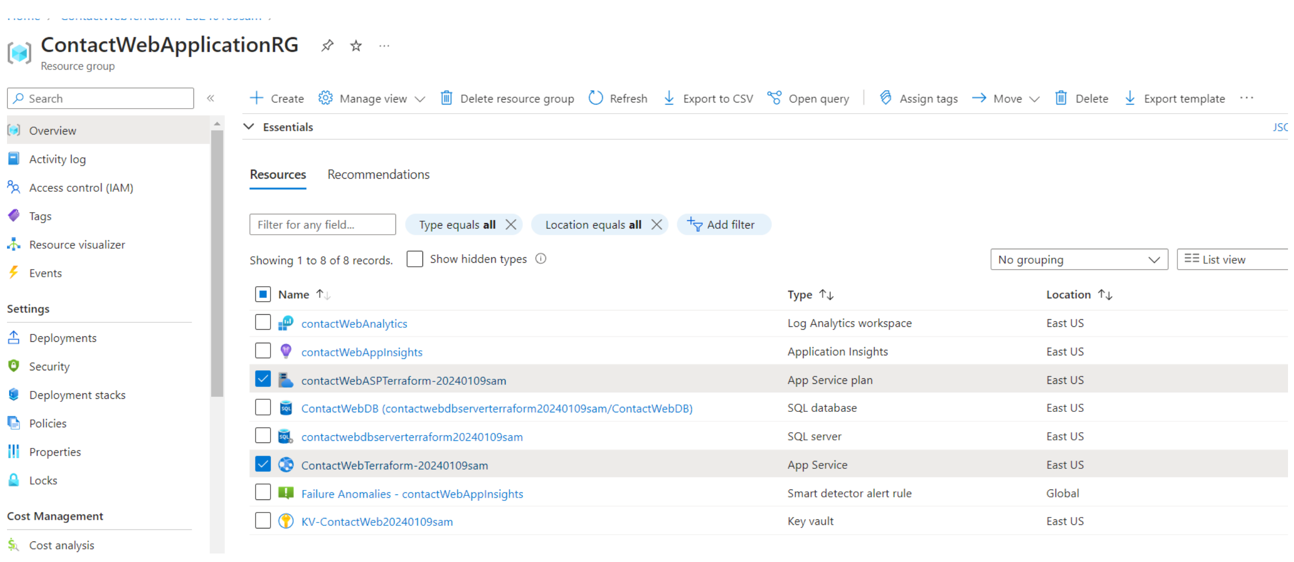
And add the missing parameters to the top of the file:

```
param webAppName string
param appServicePlanName string
param appServicePlanSku string
param identityDBConnectionStringKey string
param managerDBConnectionStringKey string
param appInsightsConnectionStringKey string
```

Aaaand... You guessed it, you need to add the new variables to the `variables.tf` in the root module.

1. Check in and deploy

   Navigate back to the resource group and review the resources.



ContactWebApplicationRG
Resource group

+ Create | Manage view | Delete resource group | Refresh | Export to CSV | Open query | Assign tags | Move | Delete | Export template

Overview | Activity log | Access control (IAM) | Tags | Resource visualizer | Events
Settings: Deployments | Security | Deployment stacks | Policies | Properties | Locks
Cost Management: Cost analysis

Essentials

Resources | Recommendations

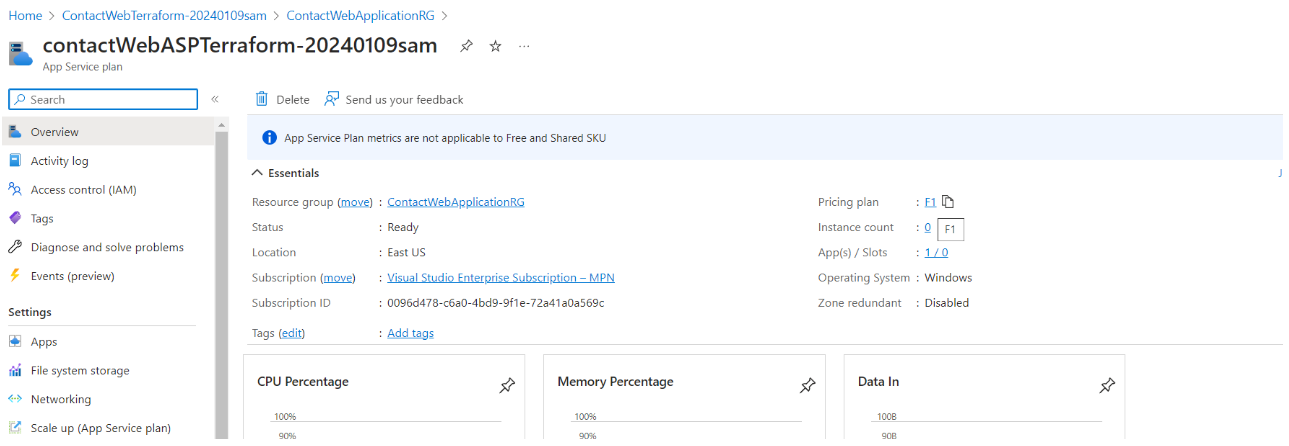
Filter for any field... | Type equals **all** | Location equals **all** | Add filter

Showing 1 to 8 of 8 records. Show hidden types | No grouping | List view

Name	Type	Location
contactWebAnalytics	Log Analytics workspace	East US
contactWebAppInsights	Application Insights	East US
contactWebASPTerraform-20240109sam	App Service plan	East US
ContactWebDB (contactwebdbserverterraform20240109sam/ContactWebDB)	SQL database	East US
contactwebdbserverterraform20240109sam	SQL server	East US
ContactWebTerraform-20240109sam	App Service	East US
Failure Anomalies - contactWebAppInsights	Smart detector alert rule	Global
KV-ContactWeb20240109sam	Key vault	East US

2. Ensure the web app is created:

   Review the App Service Plan for free tier:



Home > ContactWebTerraform-20240109sam > ContactWebApplicationRG >

contactWebASPTerraform-20240109sam
App Service plan

Delete | Send us your feedback

App Service Plan metrics are not applicable to Free and Shared SKU

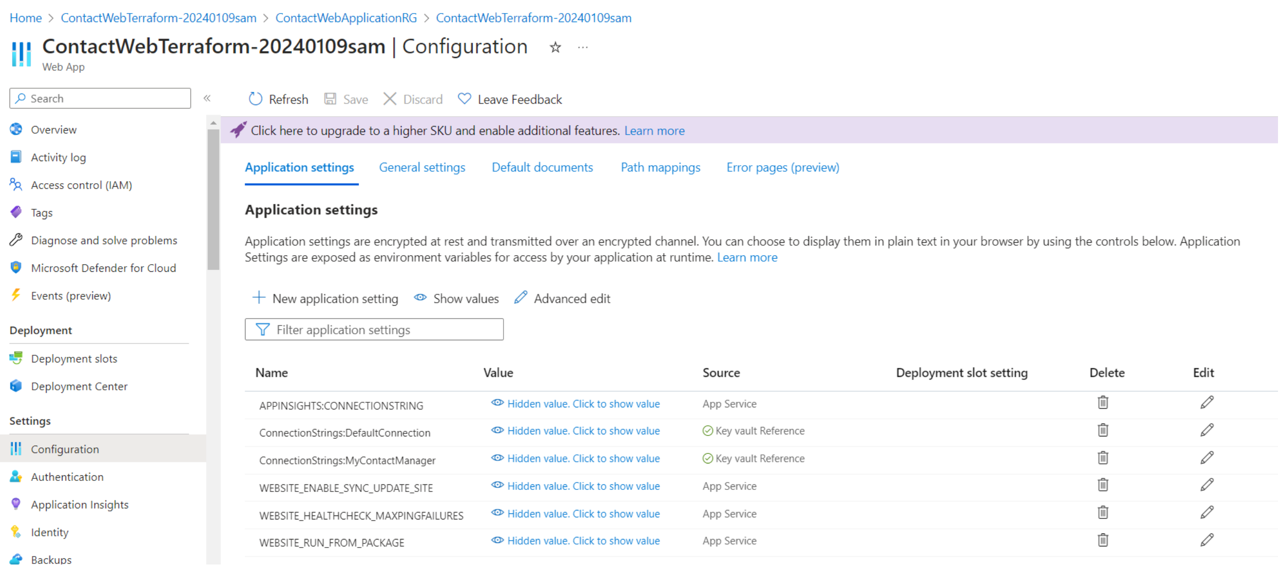
Essentials

Resource group (move)	ContactWebApplicationRG
Status	Ready
Location	East US
Subscription (move)	Visual Studio Enterprise Subscription – MPN
Subscription ID	0096d478-c6a0-4bd9-9f1e-72a41a0a569c
Tags (edit)	Add tags
Pricing plan	F1
Instance count	0
App(s) / Slots	1 / 0
Operating System	Windows
Zone redundant	Disabled

Overview | Activity log | Access control (IAM) | Tags | Diagnose and solve problems | Events (preview)
Settings: Apps | File system storage | Networking | Scale up (App Service plan)

CPU Percentage | Memory Percentage | Data In

   Review the App Service for settings, just like with the Bicep deployments, make sure that the key vault references have a green checkmark:



Home > ContactWebTerraform-20240109sam > ContactWebApplicationRG > ContactWebTerraform-20240109sam

ContactWebTerraform-20240109sam | Configuration
Web App

Refresh | Save | Discard | Leave Feedback

Click here to upgrade to a higher SKU and enable additional features. Learn more

Application settings | General settings | Default documents | Path mappings | Error pages (preview)

Application settings

Application settings are encrypted at rest and transmitted over an encrypted channel. You can choose to display them in plain text in your browser by using the controls below. Application Settings are exposed as environment variables for access by your application at runtime. Learn more

+ New application setting | Show values | Advanced edit

Filter application settings

Name	Value	Source	Deployment slot setting	Delete	Edit
APPINSIGHTS:CONNECTIONSTRING	Hidden value. Click to show value	App Service			
ConnectionStrings:DefaultConnection	Hidden value. Click to show value	✓ Key vault Reference			
ConnectionStrings:MyContactManager	Hidden value. Click to show value	✓ Key vault Reference			
WEBSITE_ENABLE_SYNC_UPDATE_SITE	Hidden value. Click to show value	App Service			
WEBSITE_HEALTHCHECK_MAXPINGFAILURES	Hidden value. Click to show value	App Service			
WEBSITE_RUN_FROM_PACKAGE	Hidden value. Click to show value	App Service			

Overview | Activity log | Access control (IAM) | Tags | Diagnose and solve problems | Microsoft Defender for Cloud | Events (preview)
Deployment: Deployment slots | Deployment Center
Settings: Configuration | Authentication | Application Insights | Identity | Backups

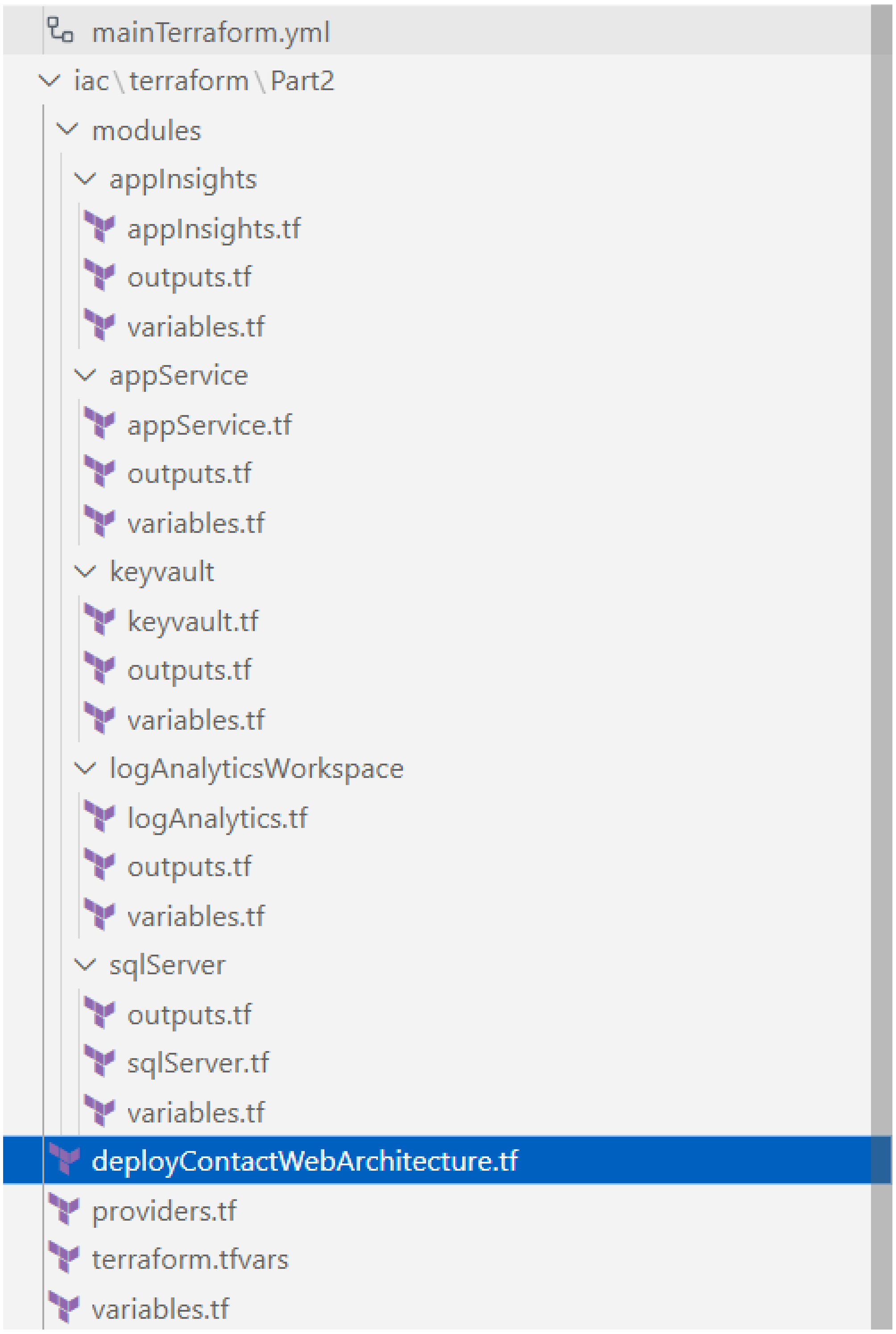
Completion Check

Make sure that the following resources and connections are created:

At the end of this step, you should have the following resources:

- Resource Group
- Log Analytics Workspace
- Application Insights
- App Service Plan
  - F1 Tier for free deployment
- App Service
  - leverages application insights instrumentation key/connection string
- Azure SQL Server
  - basic tier ($5/month)
- Key Vault
  - secret for database connection string
  - permission for app service to get/list secrets

The files as they exist in the repository should look similar to this:



```
mainTerraform.yml
iac \ terraform \ Part2
  modules
    appInsights
      appInsights.tf
      outputs.tf
      variables.tf
    appService
      appService.tf
      outputs.tf
      variables.tf
    keyvault
      keyvault.tf
      outputs.tf
      variables.tf
    logAnalyticsWorkspace
      logAnalytics.tf
      outputs.tf
      variables.tf
    sqlServer
      outputs.tf
      sqlServer.tf
      variables.tf
  deployContactWebArchitecture.tf
  providers.tf
  terraform.tfvars
  variables.tf
```

- deployContactWebArchitecture.tf
- variables.tf
- providers.tf
- terraform.tfvars

You should also have the following modules:

- appInsights
- appService
- keyVault
- logAnalyticsWorkspace
- sqlServer

Congratulations! You have completed the infrastructure deployment for the application. You can now move on to the next part of the walkthrough to deploy the application code into the App Service.