# Migrating your existing ARM Templates and Deployments to Bicep

Aaron Saikovski

Azure Principal Platform Engineer

e. asaikovski@outlook.com

t. @RuskyDuck72

Aaron Saikovski

Azure Principal Platform Engineer

*Former Microsoft CSA-E - Watch this space* ☺

Email: asaikovski@outlook.com

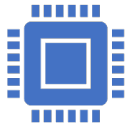
Twitter: @RuskyDuck72

GitHub: https://github.com/aaronsaikovski

LinkedIn:

## Background

# Agenda



Infrastructure as Code – Overview & Benefits

Bicep – Overview

Azure Resource Manager – Recap

Demos!

Takeaways & Resources

Q & A

# Infrastructure as Code, what is it?

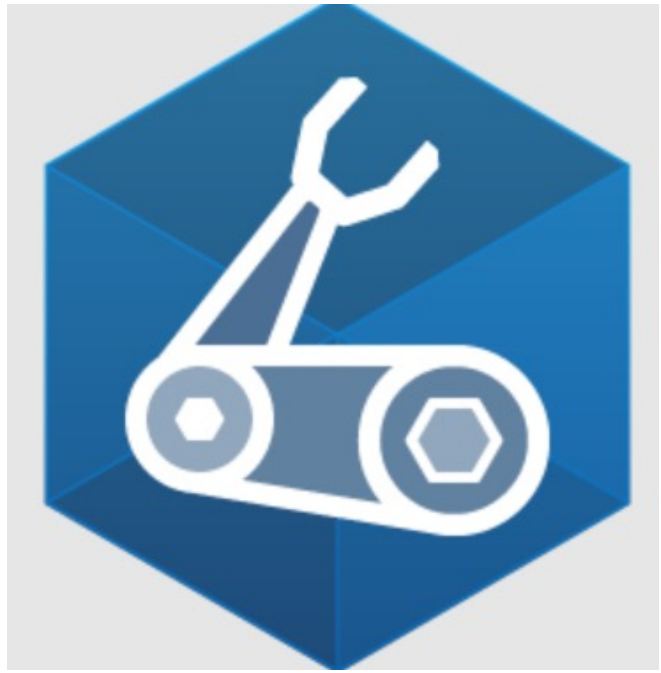
- Represents infrastructure in a:
  - Repeatable
  - Human readable text-based format
  - Modular
  - Declarative (representing the target state)
  - Automated way

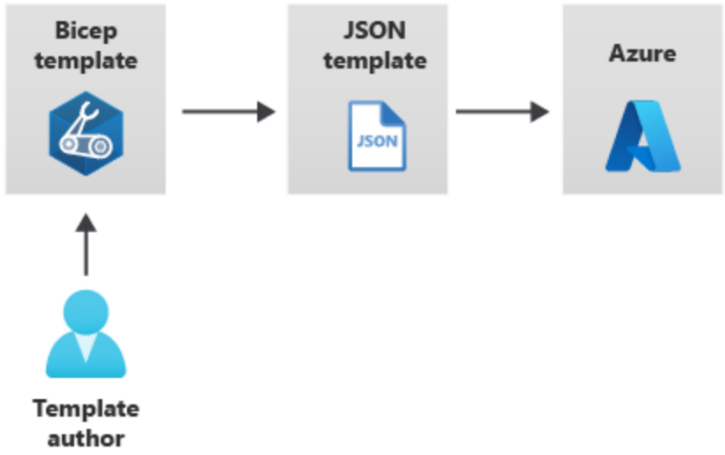
# What are the benefits?

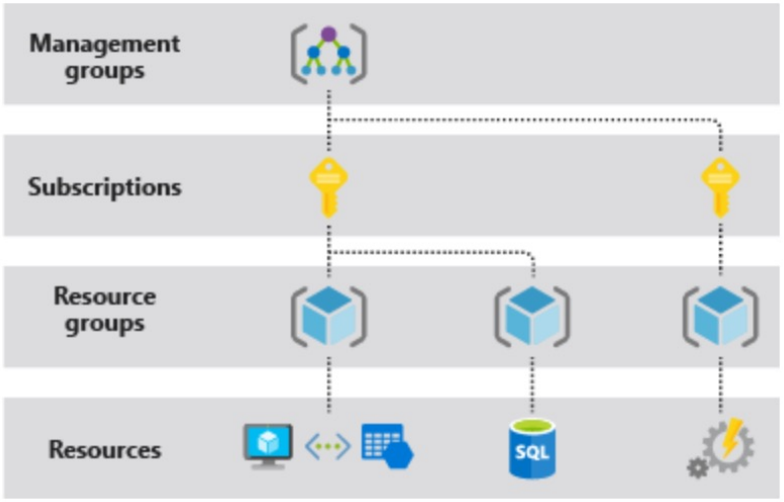
- Higher quality changes
- Allows for standard approved patterns – WAF/CAF
- Distributed changes
- Reduced permission requirements
- Less complexity to maintain
- Standardised naming conventions
- Easier to test
- Software based infrastructure deployments

# What is Bicep?

- Resource Manager template language that's used to declaratively deploy Azure resources.
- Intended to be easy to understand and straightforward to learn, regardless of your experience with other programming languages.
- Bicep transpiles into a JSON template prior to submitting to Azure resource manager.

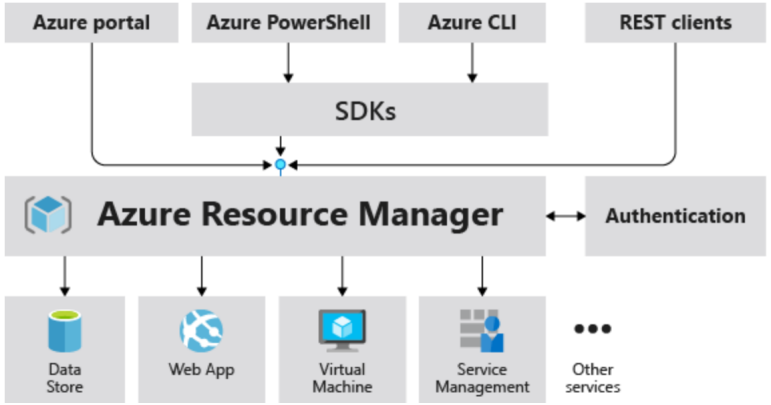







# Azure Resource Manager - Recap

- Deployment and management service for Azure
- Interact using many tools, APIs or SDKs including the Azure portal
- **Control** and **Data** plane operations. Use the control plane to manage the resources in your subscription. Use the data plane to access features that are exposed by a resource.



# Demos

Migration To/From -> Bicep/ARM

Using Modules – local and remote

Validation and testing

# Key Takeaways

1. Keep your Bicep version updated – version 0.17.1 – updated monthly
2. Use Modules – encapsulation, reusability, predictability
3. Test your code – GitHub Actions
4. Validate your code locally – Pre-Commit hooks
5. Use PSRule to analyse your code (*Ask Bernie White all about it* ☺ )
6. Don't hardcode any parameters…set defaults..**no passwords**!!!
7. Use '**main.bicep**' as your main template per stack/deployment
8. Use "`what-if`" to evaluate changes between deployments
9. Unit test your Bicep code - Benchpress & others
10. Follow Microsoft best practices for developing Bicep templates
11. Visual Studio Code is your friend..its free!
12. Automate all the things

# Resources

1. Bicep Releases - https://github.com/Azure/bicep/releases
2. Common Azure Resource Modules Library (CARML) - https://aka.ms/carml/
3. GitHub Actions - https://learn.microsoft.com/en-au/training/modules/test-bicep-code-using-github-actions/
4. Bicep PreCommit hooks - https://github.com/Azure4DevOps/check-azure-bicep
5. PSRule - https://azure.github.io/PSRule.Rules.Azure/using-bicep/
6. Benchpress Unit Testing Framework - https://github.com/azure/benchpress
7. Microsoft best practices for developing Bicep templates: https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/best-practices
8. Bicep Community Call - Sign up: https://aka.ms/armnews

# Thank you & Q/A