Table of Contents

[Introduction 5](#_Toc81565010)

[Redis 5](#_Toc81565011)

[Azure Cache for Redis 5](#_Toc81565012)

[Use Case 6](#_Toc81565013)

[Overview 6](#_Toc81565014)

[Assessment 8](#_Toc81565015)

[Redis Versions 8](#_Toc81565016)

[Architecture and Objects 9](#_Toc81565017)

[Limitations 9](#_Toc81565018)

[Databases 9](#_Toc81565019)

[Source Systems 10](#_Toc81565020)

[Performance Analysis Tools 10](#_Toc81565021)

[Azure Cache for Redis - Service Tiers 10](#_Toc81565022)

[WWI Use Case 13](#_Toc81565023)

[Assessment Checklist 13](#_Toc81565024)

[Planning 14](#_Toc81565025)

[Landing Zone 14](#_Toc81565026)

[Networking 14](#_Toc81565027)

[Private Link and/or VNet integration 15](#_Toc81565028)

[SSL/TLS Connectivity 15](#_Toc81565029)

[WWI Use Case 15](#_Toc81565030)

[Planning Checklist 16](#_Toc81565031)

[Migration Methods 16](#_Toc81565032)

[Import / Export 16](#_Toc81565033)

[Fastest/Minimum Downtime Migration 18](#_Toc81565034)

[Decision Table 18](#_Toc81565035)

[WWI Use Case 18](#_Toc81565036)

[Migration Methods Checklist 18](#_Toc81565037)

[Test Plans 19](#_Toc81565038)

[Overview 19](#_Toc81565039)

[Sample Queries 19](#_Toc81565040)

[Rollback Strategies 20](#_Toc81565041)

[WWI Use Case 20](#_Toc81565042)

[Checklist 20](#_Toc81565043)

[Performance Baselines 21](#_Toc81565044)

[Tools 21](#_Toc81565045)

[redis-benchmark 21](#_Toc81565046)

[Server Configuration 21](#_Toc81565047)

[WWI Use Case 21](#_Toc81565048)

[Data Migration with Backup and Restore (RDB) 22](#_Toc81565049)

[Setup 22](#_Toc81565050)

[Disable AOF in Target 22](#_Toc81565051)

[Data 22](#_Toc81565052)

[Enable AOF in Target 24](#_Toc81565053)

[Summary - Backup and Restore 25](#_Toc81565054)

[Resources 25](#_Toc81565055)

[Setup 25](#_Toc81565056)

[Connectivity Setup 25](#_Toc81565057)

[Disable AOF in Target 25](#_Toc81565058)

[Mass Insertion (SET) 25](#_Toc81565059)

[Mass Insertion (DUMP/RESTORE) 26](#_Toc81565060)

[Mass Insertion (MIGRATE) 27](#_Toc81565061)

[Enable AOF in Target 27](#_Toc81565062)

[Data Migration with Redis Replication 27](#_Toc81565063)

[Setup 28](#_Toc81565064)

[Option 1 : Set to Read Only 28](#_Toc81565065)

[Option 2 : Rename Commands 28](#_Toc81565066)

[Replication (SLAVEOF) : Pre Redis 5.0 28](#_Toc81565067)

[Replication (REPLICAOF) : 5.x or higher 29](#_Toc81565068)

[Resources 29](#_Toc81565069)

[Setup 29](#_Toc81565070)

[Install STunnel 29](#_Toc81565071)

[Disable AOF in Target 29](#_Toc81565072)

[Redis-migrate 29](#_Toc81565073)

[redis-copy (yaauie) 31](#_Toc81565074)

[redis-copy (deepakverma) 31](#_Toc81565075)

[redis-dump 32](#_Toc81565076)

[redis-dump 32](#_Toc81565077)

[Enable AOF in Target 32](#_Toc81565078)

[Resources 32](#_Toc81565079)

[Example 32](#_Toc81565080)

[Data Migration 33](#_Toc81565081)

[Back up the instance 33](#_Toc81565082)

[Offline vs. Online 33](#_Toc81565083)

[Data Drift 33](#_Toc81565084)

[Performance recommendations 33](#_Toc81565085)

[Performing the Migration 34](#_Toc81565086)

[Common Steps 35](#_Toc81565087)

[WWI Use Case 35](#_Toc81565088)

[instance Objects 35](#_Toc81565089)

[Execute migration 35](#_Toc81565090)

[Data Migration Checklist 36](#_Toc81565091)

[Data Migration - Configuration 36](#_Toc81565092)

[Configuration 36](#_Toc81565093)

[Data Migration - Application Settings 38](#_Toc81565094)

[Setup 38](#_Toc81565095)

[Update Applications to support SSL 38](#_Toc81565096)

[Redeploy the application 38](#_Toc81565097)

[Test the application 38](#_Toc81565098)

[Post Migration Management 39](#_Toc81565099)

[Monitoring and Alerts 39](#_Toc81565100)

[Server Configuration 40](#_Toc81565101)

[PowerShell Module 40](#_Toc81565102)

[Azure Cache for Redis Upgrade Process 40](#_Toc81565103)

[WWI Use Case 41](#_Toc81565104)

[Management Checklist 41](#_Toc81565105)

[Optimization 41](#_Toc81565106)

[Monitoring Hardware and Cache Performance 41](#_Toc81565107)

[Typical Workloads and Performance Issues 41](#_Toc81565108)

[Azure Monitor 42](#_Toc81565109)

[Scale the Server 42](#_Toc81565110)

[Moving Regions 42](#_Toc81565111)

[Quick Tips 42](#_Toc81565112)

[WWI Use Case 43](#_Toc81565113)

[Optimization Checklist 43](#_Toc81565114)

[Business Continuity and Disaster Recovery (BCDR) 43](#_Toc81565115)

[Backup and Restore 43](#_Toc81565116)

[Replicas 44](#_Toc81565117)

[Deleted Servers 44](#_Toc81565118)

[Regional Failure 45](#_Toc81565119)

[WWI Use Case 45](#_Toc81565120)

[BCDR Checklist 46](#_Toc81565121)

[Security 46](#_Toc81565122)

[Authentication 46](#_Toc81565123)

[Threat Protection 46](#_Toc81565124)

[Audit Logging 46](#_Toc81565125)

[Encryption 47](#_Toc81565126)

[Firewall 47](#_Toc81565127)

[Private Link 47](#_Toc81565128)

[Security baseline 48](#_Toc81565129)

[Security Checklist 48](#_Toc81565130)

[Summary 48](#_Toc81565131)

[Questions and Feedback 48](#_Toc81565132)

[Find a partner to assist in migrating 49](#_Toc81565133)

[Appendix A: Environment Setup 49](#_Toc81565134)

[Deploy the ARM template 49](#_Toc81565135)

[Setup Redis Binding 51](#_Toc81565136)

[Setup Redis Cluster 51](#_Toc81565137)

[Open the Azure VM Ports 51](#_Toc81565138)

[Allow Azure Redis Access 52](#_Toc81565139)

[Connect to the Azure VM 52](#_Toc81565140)

[Install Redis 6.x 52](#_Toc81565141)

[Download artifacts 54](#_Toc81565142)

[Deploy the instance 54](#_Toc81565143)

[Configure the Web Application API 54](#_Toc81565144)

[Test the Web Application 56](#_Toc81565145)

[Configure the Web Application Client 56](#_Toc81565146)

[Deploy the Java Server Application to Azure 57](#_Toc81565147)

[Deploy the Angular Web Application to Azure 60](#_Toc81565148)

[Configure Network Security (Secure path) 63](#_Toc81565149)

[Appendix B: ARM Templates 63](#_Toc81565150)

[Secure 63](#_Toc81565151)

[Non-Secure 63](#_Toc81565152)

[SSL Connectivity with STunnel 63](#_Toc81565153)

[Install STunnel 63](#_Toc81565154)

[Configure Redis SSL 64](#_Toc81565155)

[Download Redis 6.0 64](#_Toc81565156)

[Build with TLS support 64](#_Toc81565157)

[Create Certificates 65](#_Toc81565158)

[Start the server 65](#_Toc81565159)

[Test connectivity 65](#_Toc81565160)

[Resources 65](#_Toc81565161)

# Introduction

This migration guide is designed to provide snackable and actionable information for Redis customers seeking to migrate on-premises or non-Azure hosted Redis workloads to [Azure Cache for Redis](https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-overview). This guide will provide realistic guidance planning for executing the migration of a majority of Redis configurations to Azure.

Workload functionality and existing application connectivity can present challenges when migrating existing Redis instances to the cloud. This guide offers helpful links and recommendations focusing on migrating the cache, ensuring performance, and functional application operational parity.

The information provided will center on a customer journey using the Microsoft [Cloud Adoption Framework](https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/get-started/) to perform assessment, migration, and post-optimization activities for an Azure Cache for Redis environment.

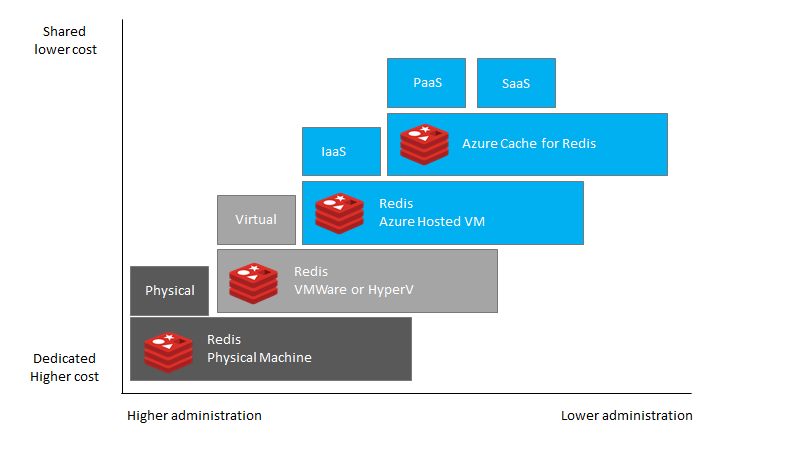
## Redis

Redis has a rich history in the open source community and is heavily used in corporate websites and critical applications. This guide will assist administrators who have been asked to scope, plan, and execute the migration. Administrators new to Redis can also review the [Redis Documentation](https://redis.io/documentation) for deeper information of the internal workings on Redis. Additionally, each document section contains links to helpful reference articles and tutorials.

## Azure Cache for Redis

Microsoft offers a fully managed Redis cache environment to support your data cache, content cache, session store and many other applications as a Platform as a Service (PaaS) offering. In this fully managed environment, the operating system and software updates are automatically applied, as well as the implementation of high availability and protection of the data. For more overview about the Redis cache offering in Azure, reference the [About Azure Cache for Redis](https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-overview) documentation page.

Although we will solely focus on the managed service, Redis can also run in Azure Virtual machines..



Comparison of Redis environments.

This guide will focus entirely on migrating the on-premises Redis workloads to the Platform as a Service (PaaS) Azure Cache for Redis offering due to its various advantages over Infrastructure as a Service (IaaS) such as scale-up and scale-out, pay-as-you-go, high availability, security and manageability features.

# Use Case

## Overview

World Wide Importers (WWI) is a San Francisco, California-based manufacturer and wholesale distributor of novelty goods. They began operations in 2002 and developed an effective business-to-business (B2B) model, selling the items they produce directly to retail customers throughout the United States. Its customers include specialty stores, supermarkets, computing stores, tourist attraction shops, and some individuals. This B2B model enables a streamlined distribution system of their products, allowing them to reduce costs and offer more competitive pricing on the items they manufacture. They also sell to other wholesalers via a network of agents who promote their products on WWI’s behalf.

Before launching into new areas, WWI wants to ensure its IT infrastructure can handle the expected growth. WWI currently hosts all its IT infrastructure on-premises at its corporate headquarters and believes moving these resources to the cloud enables future growth. They have tasked their CIO with overseeing the migration of their customer portal and the associated data workloads to the cloud.

WWI would like to continue to take advantage of the many advanced capabilities available in the cloud, and they are interested in migrating their instances and associated workloads into Azure. They want to do this quickly and without having to make any changes to their applications or instances. Initially, they plan on migrating their java-based customer portal web application and the associated Redis instances and workloads to the cloud.

### Migration Goals

The primary goals for migrating their Redis instances and associated workloads to the cloud include:

* Improve their overall security posture by encrypting data at rest and in-transit.
* Enhance the high availability and disaster recovery (HA/DR) capabilities.
* Position the organization to leverage cloud-native capabilities and technologies such as point in time restore.
* Take advantage of administrative and performance optimizations features of Azure Cache for Redis.
* Create a scalable platform that they can use to expand their business into more geographic regions.
* Allow for enhanced compliance with various legal requirements where PII information is stored.

WWI used the [Cloud Adoption Framework (CAF)](https://docs.microsoft.com/azure/cloud-adoption-framework/) to educate their team on following best practices guidelines for cloud migration. Using CAF as a higher-level migration guide, WWI customized their migration into three main stages. Within each stage, they defined activities that needed to be addressed to ensure a successful lift and shift cloud migration.

These stages include:

|  |  |  |
| --- | --- | --- |
| Stage | Name | Activities |
| 1 | Pre-migration | Assessment, Planning, Migration Method Evaluation, Application Implications, Test Plans, Performance Baselines |
| 2 | Migration | Execute Migration, Execute Test Plans |
| 3 | Post-migration | Business Continuity, Disaster Recovery, Management, Security, Performance Optimization, Platform modernization |

WWI has several instances of Redis running with varying versions ranging from 3.0 to 6.0. They would like to move their older instances to the latest Redis version as soon as possible, but there are some concerns regarding applications functioning without issues. A decision has been made to migrate to the cloud first and upgrade the Redis version later knowing that Redis 3.0 and 4.0 are coming to end of support.

They would also like to ensure that their data workloads are safe and available across multiple geographic regions in case of failure and are looking at the available configuration options.

WWI wants to start off with a simple application for the first migration, and then move to more business-critical applications in a later phase. This will provide the team with the knowledge and experience they need to prepare and plan for those future migrations.

# Assessment

Before jumping right into migrating a Redis workload, there is a fair amount of due diligence that must be performed. This includes analyzing the data, hosting environment, and application workloads to validate the Azure landing zone is properly configured and prepared to host the soon-to-be migrated workloads.

## Redis Versions

Remote Dictionary Server (Redis) has a rich history starting in late 2000s. Since then, it has evolved into a widely used memory based key value data management system. Azure Cache for Redis started with the support of Redis version 4.0 and has continued to 6.0 (as of 8/2021). For a listing of all Redis versions, reference [this detailed page](https://bucardo.org/postgres_all_versions.html).

For the latest on Azure Cache for Redis version support, reference [Supported Azure Cache for Redis server versions](https://docs.microsoft.com/en-us/azure/Redis/concepts-supported-versions). In the Post Migration Management section, we will review how upgrades (such as 4.0 to 6.0) are applied to the Redis instances in Azure.

**Note** Redis Support is based on the release of the latest stable release. Only the latest stable version, and the last two versions will receive maintenance. As of 4/2021, anything prior to 4.0 is end of life.

Knowing the source Redis version is important as many features have been introduced through the major versions. The applications using the system may be expecting behavior and features that are specific to that version. Although Redis has been great at keeping breaking changes at a minimum and keeping compatibility between versions, there are a small handful of cases that may influence the migration path and version:

* Support for Redis Modules (RedisSearch, RedisBloom, RedisTimeSeries)
* RDB format change (5.0 not backwards compatible, 4.0 is not backwards compatible)
* Change in INFO fields (4.0)
* Usage of REDIS ACL (6.0+)
* RESP3 mode (6.0+)
* LRU Cache changes (4.0+)
* Any Lua Language changes (EVAL, EVALSHA)
* Extensive use of TTL
* Number of databases

To check the Redis server version run the following command against the Redis instance:

redis-server --version

For a list of changes between versions, reference the latest release documentation:

* [Redis 6.x](https://raw.githubusercontent.com/redis/redis/6.0/00-RELEASENOTES)
* [Redis 5.x](https://raw.githubusercontent.com/redis/redis/5.0/00-RELEASENOTES)
* [Redis 4.x](https://raw.githubusercontent.com/redis/redis/4.0/00-RELEASENOTES)
* [Redis 3.x](https://raw.githubusercontent.com/redis/redis/3.0/00-RELEASENOTES)

## Architecture and Objects

Data (keys and the values) is only one component of instance migration. The instance supporting configuration may also need to be migrated and validated to ensure the applications will continue to run reliably. As part of the assessment, it is important to understand what features of the system are being used other than data storage.

Here is a list of inventory items that should be inventoried before and after the migration:

* Users
* Configuration settings

After reviewing the above items, notice there is much more than just data that may be required to migrate a Redis workload. The following sections below address more specific details about several of the above.

## Limitations

Azure Cache for Redis is a fully supported version of Redis running as a platform as a service. However, there are some common limitations to become familiar with when doing an initial assessment for selected your landing zone. Many of these limitations are driven by the tier selected as shown in the online [supported features document](https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-overview#feature-comparison)

Many of the other items are simply operational aspects that administrators should become familiar with as part of the operational data workload lifecycle management. This guide will explore many of these operational aspects in the [Post Migration Management](file:///D:\data\projects\redis-cache\pandoc\04_PostMigration\01_Management.md) section.

**Note** Each tier supports a maximum number of databases. If you have more than the default of 16, be sure that you pick a tier to migrate too that has support for all source databases.

### Redis Modules

You can extend the features of Redis by implemented custom Redis modules. Look for any loadmodule directives in the redis.conf file that are not part of the default installation. You can also get a list of all modules by running:

redis-cli MODULE LIST

## Databases

When performing a migration, consider the Redis instance may have more than one database. The tool you select will need to be able to support migrating all databases versus just the default 0 database. You can find the number of databases by running the following:

redis-cli INFO keyspace

## Source Systems

The amount of migration preparation can vary depending on the source system and its location. In addition to the instance objects, consider how to get the data from the source system to the target system. Migrating data can become challenging when there are firewalls and other networking components in between the source and target.

Internet migration speed is an important factor. Moving data over the Internet can be slower than using dedicated circuits to Azure. Consider setting up an [ExpressRoute](https://docs.microsoft.com/en-us/azure/expressroute/expressroute-introduction) connection between the source network and the Azure network when moving many gigabytes, terabytes, and petabytes of data.

Do not overwhelm existing network infrastructure. If ExpressRoute is already present, the connection is likely being used by other applications. Performing a migration over an existing route can cause strain on the network throughput and potentially cause considerable performance degradation for both the migration and other applications using the network.

Lastly, disk space must be evaluated. When exporting a very large instance, consider the size of the data. Ensure the system where the tool is running and the export location have enough disk space to perform the export operation.

### Cloud Providers

Migrating instances from cloud services providers, such as Google Cloud (GCP) and Amazon Web Services (AWS), may require extra networking configuration steps to access the cloud-hosted Redis instances or they may prevent Redis migration commands. Any first party or third party migration tools will require access from outside IP ranges and may be blocked by default.

### On-premises

Like cloud provider-hosted environments, if the Redis data environment is behind corporate firewalls or other network security layers, a path will need to be created between the on-premises instance and Azure Cache for Redis.

## Performance Analysis Tools

Many tools and methods can be used to assess the Redis data workloads and environments. Each tool will provide a different set of assessment and migration features and functionality. As part of this guide, we will review the most commonly used tools for assessing Redis data workloads.

## Azure Cache for Redis - Service Tiers

Equipped with the assessment information (CPU, memory, storage, etc.), the migration user’s next choice is to decide which Azure Cache for Redis service and pricing tier to start with.

There are currently five potential options:

* Azure Cache for Redis (Basic) : An OSS Redis cache running on a single VM. This tier has no service-level agreement (SLA) and is ideal for development/test and non-critical workloads.
* Azure Cache for Redis (Standard) : An OSS Redis cache running on two VMs in a replicated configuration.
* Azure Cache for Redis (Premium) : High-performance OSS Redis caches. This tier offers higher throughput, lower latency, better availability, and more features. Premium caches are deployed on more powerful VMs compared to the VMs for Basic or Standard caches.
* Azure Cache for Redis (Enterprise) : High-performance caches powered by Redis Labs’ Redis Enterprise software. This tier supports Redis modules including RediSearch, RedisBloom, and RedisTimeSeries. Also, it offers even higher availability than the Premium tier.
* Azure Cache for Redis (Enterprise Flash) : Cost-effective large caches powered by Redis Labs’ Redis Enterprise software. This tier extends Redis data storage to non-volatile memory, which is cheaper than DRAM, on a VM. It reduces the overall per-GB memory cost.

Briefly, these options were discussed in the [Limitations](#Xef4603c8647a4d6019c31c84ecbbd1d46b17e4e) document.

### Comparison of Services

Which Azure Cache for Redis service should be selected and used? This table outlines some of the advantages and disadvantages of each along with their Redis version support as of 4/2021.

|  |  |  |  |
| --- | --- | --- | --- |
| Service | Pros | Cons | Versions Supported |
| Azure VM | Any version, most flexible, full Redis feature support | Customer responsible for updates, security, and administration | Any Version |
| Basic | Sizes up to 53GB, low cost | Lower performance, no data persistence, no replication or failover | 4.x, 6.x |
| Standard | All basic, plus replication and failover support | Lower performance, no data persistence | 4.x, 6.x |
| Premium | All Standard, plus zone redundancy, data persistence and clustering | No support for Redis Modules | 4.x, 6.x |
| Enterprise | All Premium, plus Redis Module support | Higher costs | 6.x |
| Enterprise Flash | Flash based memory | No Redis Module support | 6.x |

As displayed above, if the instance is running Redis 3.x or lower and do not plan to upgrade, the workload will need to run in an Azure VM.

### Costs

After evaluating the entire WWI Redis data workloads, WWI determined they would need at least 6GB of cache capacity with data persistence and clustering support so a Premium Sku was selected. WWI intentionally chose to begin its Azure migration journey with a relatively small workload. However, the best practices of instance migration still apply and will be used as a template for future migrations.

To determine the memory usage, they interrogated the redis processes on their source system during a heavy load period:

ps -o pid,user,%mem,command ax | sort -b -k3 -r

They then monitored the network bandwidth to see how much traffic was being used between the clients and the Redis server. They measured about 15% cache usage per hour which equated to 900MB of traffic per hour which equates to 328GB of traffic per year. The current application will not be moved to the same Azure region but will utilize the Azure Redis Cache which means network bandwidth will have to be paid. They had a couple of tools to choose from to monitor the network traffic (iptraf and nethogs):

sudo apt-get install iptraf -y  
  
sudo netstat =tump | grep <port\_number>

sudo apt-get install nethogs  
  
sudo nethogs

Additionally, because they want data persistence and backups, they will persist this to Azure Storage.

Using the [Azure Cache for Redis pricing calculator](https://azure.microsoft.com/en-us/pricing/details/cache/) WWI was able to determine the costs for the Azure Cache for Redis instance. As of 8/2021, the total costs of ownership (TCO) is displayed in the following table for the WWI Conference instance:

|  |  |  |  |
| --- | --- | --- | --- |
| Resource | Description | Quantity | Cost |
| Compute (Premium) | 6GB Memory (1 primary, 1 replica) | 24 x 365 @ $0.554/hr | $4853.04 / yr |
| Storage (backup) | 6GB | 6 \* 12 @ $0.15 | $10.80 / yr |
| Network | ~27.37GB/month egress | 12 \* 22.37 \* $.08 | $21.4752 / yr |
| Total |  |  | $4885.31 / yr |

After reviewing the initial costs, WWI’s CIO confirmed they will be on Azure for a period much longer than 3 years. They decided to use 3-year [reserve instances](https://docs.microsoft.com/en-us/azure/Redis/concept-reserved-pricing) to save an extra ~$2.6K/yr:

|  |  |  |  |
| --- | --- | --- | --- |
| Resource | Description | Quantity | Cost |
| Compute (Premium) | 6GB Memory (1 primary, 1 replica) | 24 x 365 @ $0.249/hr | $2190 / yr |
| Storage (backup) | 6GB | 6 \* 12 @ $0.15 | $10.80 / yr |
| Network | ~27.37GB/month egress | 12 \* 22.37 \* $.08 | $21.4752 / yr |
| Total |  |  | $2222.27 / yr (~45% savings) |

As the table above shows, backups, network egress, and any extra nodes must be considered in the total cost of ownership (TCO). As more instances are added, the storage and network traffic generated would be the only extra cost-based factor to consider.

**Note:** The estimates above do not include any [ExpressRoute](https://docs.microsoft.com/en-us/azure/expressroute/expressroute-introduction), [Azure App Gateway](https://docs.microsoft.com/en-us/azure/application-gateway/overview), [Azure Load Balancer](https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-overview), or [App Service](https://docs.microsoft.com/en-us/azure/app-service/overview) costs for the application layers.

The above pricing can change at any time and will vary based on region. The region used above was West US 2.

### Application Implications

When moving to Azure Cache for Redis, the conversion to secure sockets layer (SSL) based communication is likely to be one of the biggest changes for the applications. SSL is enabled by default in Azure Cache for Redis and it is likely the on-premises application and data workload is not set up to connect to Redis using SSL. When enabled, SSL usage will add some additional processing overhead and should be monitored.

**Note** Although SSL is enabled by default, it is possible to disable. This is strongly not recommended.

Follow the activities in [Configure TLS connectivity in Azure Cache for Redis](https://docs.microsoft.com/en-us/azure/Redis/concepts-ssl-connection-security) to reconfigure the application to support this strong authentication path.

Lastly, modify the server name in the application connection strings or switch the DNS to point to the new Azure Cache for Redis server.

## WWI Use Case

WWI started the assessment by gathering information about their Redis instances. They were able to compile the following:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Source | Size | Data Persistence | Version | Owner | Downtime |
| Redis (Www) | AWS (PaaS) | 6GB | yes | 3.0 | Information Technology | 4 hr |
| Redis (Database) | On-premises | 12GB | yes | 5.0 | Information Technology | 1 hrs |

Each instance owner was contacted to determine the acceptable downtime period. The planning and migration method selected was based on the acceptable instance downtime.

For the first phase, WWI focused solely on the web site supporting instance. The team needed the migration experience to assist in the proceeding data workload migrations. The www instance was selected because of the simple instance structure and the lenient downtime requirements. Once the instance was migrated, the team focused on migrating the application into the secure Azure landing zone.

## Assessment Checklist

* Test the workload runs successfully on the target system.
* Ensure the right networking components are in place for the migration.
* Understand the data workload resource requirements.
* Estimate the total costs.
* Understand the downtime requirements.
* Be prepared to make application changes.

# Planning

## Landing Zone

An [Azure Landing zone](https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/landing-zone/) is the target environment defined as the final resting place of a cloud migration project. In most projects, the landing zone should be scripted via ARM templates for its initial setup. Finally, it should be customized with PowerShell or the Azure Portal to fit the needs of the workload.

Since WWI is based in San Francisco, all resources for the Azure landing zone were created in the US West 2 region. The following resources were created to support the migration:

* [Azure Cache for Redis](https://docs.microsoft.com/en-us/azure/Redis/quickstart-create-Redis-server-instance-using-azure-portal)
* [Express Route](https://docs.microsoft.com/en-us/azure/expressroute/expressroute-introduction)
* [Azure Virtual Network](https://docs.microsoft.com/en-us/azure/virtual-network/quick-create-portal) with [hub and spoke design](https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/hybrid-networking/hub-spoke) with corresponding [virtual network peerings](https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-peering-overview) establish.
* [App Service](https://docs.microsoft.com/en-us/azure/app-service/overview)
* [Application Gateway](https://docs.microsoft.com/en-us/azure/load-balancer/quickstart-load-balancer-standard-internal-portal?tabs=option-1-create-internal-load-balancer-standard)
* [Private endpoints](https://docs.microsoft.com/en-us/azure/private-link/private-endpoint-overview) for the App Services and Redis instance

**Note:** As part of this guide, two ARM templates (one with private endpoints, one without) were provided to deploy a potential Azure landing zone for a Redis migration project. The private endpoints ARM template provides a more secure, production-ready scenario. Additional manual Azure landing zone configuration may be necessary, depending on the requirements.

## Networking

Getting data from the source system to Azure Cache for Redis in a fast and optimal way is a vital component to consider in a migration project. Small unreliable connections may require administrators to restart the migration several times until a successful result is achieved. Restarting migrations due to network issues can lead to wasted effort, time and money.

Take the time to understand and evaluate the network connectivity between the source, tool, and destination environments. In some cases, it may be appropriate to upgrade the internet connectivity or configure an ExpressRoute connection from the on-premises environment to Azure. Once on-premises to Azure connectivity has been created, the next step is to validate that the selected migration tool can connect from the source to the destination.

The migration tool location will determine the network connectivity requirements. As shown in the table below, the selected migration tool must be able to connect to both the on-premises machine and Azure. Azure should be configured to only accept network traffic from the migration tool location.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Migration Tool | Type | Tool Location | Inbound Network Requirements | Outbound Network Requirements |
| Import/Export (RDB) | Offline | On-premises | None | A path to copy the file to the new instance |
| DUMP/RESTORE | Online | On-premises | None | Open port to the target instance |
| SLAVEOF / REPLICAOF | Online | On-premises | None | Open port to the target instance |
| MIGRATE | Online | On-premises | None | Open port to the target instance |
| 3rd party tools | Offline  Online | On-premises | Based on tool | Based on tool |

**Note** We will discuss these migration methods in more detail in the next section.

Other networking considerations include:

* When using an Azure Virtual Machine to run the migration tools, assign it a public IP address and then only allow it to connect to the on-premises Redis instance.
* Outbound firewalls must ensure outbound connectivity to Azure Cache for Redis. The Redis gateway IP addresses are available on the [Connectivity Architecture in Azure Cache for Redis](https://docs.microsoft.com/en-us/azure/Redis/concepts-connectivity-architecture#azure-instance-for-Redis-gateway-ip-addresses) page.

## Private Link and/or VNet integration

All Azure Cache for Redis services support private links and VNet integration. There are however be sure to review the [FAQs for private endpoints](https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-private-link#faq) to understand the behavior of the cache when behind a private endpoint.

## SSL/TLS Connectivity

In addition to the application implications of migrating to SSL-based communication, the SSL/TLS connection types are also something that needs to be considered. After creating the Azure Cache for Redis instance, review the SSL settings, and read the [Configure your application to use TLS 1.2](https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-private-link#faq) article to understand how the TLS settings can affect the security posture of an application.

## WWI Use Case

WWI’s cloud team has created the necessary Azure landing zone resources in a specific resource group for the Azure Cache for Redis. Additional resources will be included to support the applications. To create the landing zone, WWI decided to script the setup and deployment using ARM templates. By using ARM templates, they would be able to quickly tear down and re-setup the environment, if needed.

As part of the ARM template, all connections between virtual networks will be configured with peering in a hub and spoke architecture. The instance and application will be placed into separate virtual networks. An Azure App Gateway will be placed in front of the app service to allow the app service to be isolated from the Internet. The Azure App Service will connect to the Azure Cache for Redis using a private endpoint.

WWI originally wanted to test an online migration, but the required network setup for DMS to connect to their on-premises environment made this infeasible. WWI chose to do an offline migration instead. The Redis pgAdmin tool was used to export the on-premises data and then was used to import the data into the Azure Cache for Redis instance. The WWI migration team has also learned that the versatile Azure Data Studio tool has preview Redis support, and would like to explore its utility for developing applications using Redis.

## Planning Checklist

* Prepare the Azure landing zone. Consider using ARM template deployment in case the environment must be torn down and rebuilt quickly.
* Verify the networking setup. Verification should include testing connectivity, bandwidth, latency, and firewall configurations.
* Determine if you are going to use the online or offline data migration strategy.
* Decide on the SSL certificate strategy.

# Migration Methods

Getting the data from the source to target will require using tools or features of Redis to accomplish the migration.

It is important to complete the entire assessment and planning stages before starting the next stages. The decisions and data collected are migration path and tool selection dependencies.

We explore the following commonly used tools in this section:

* Database export/import via RDB file
* Append Only File (AOF)
* Layer of abstraction
* SLAVEOF / REPLICAOF commands
* MIGRATE command
* 3rd Party tools

## Import / Export

### Redis Persistence

Redis is a memory server, designed for high-performance storage and retrieval. If the server or service where to be shutdown, all the items in the cache would be lost. To ensure durability, you can [select a persistence mode](https://redis.io/topics/persistence) to keep the values in the case of failure. These two persistence methods include RDB and AOF.

You can also select the [persistence in Azure Redis instances](https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-how-to-premium-persistence).

### RDB File

By default, Redis will keep cache data persisted to disk on a fairly regular basis, this can however be disabled by the administrator to improve performance. However, doing so would cause any data in memory to be lost in the case of a server fault or reboot. In most cases this is enabled.

### Append Only File (AOF)

The append-only file is an alternative to RDB and is a fully-durable strategy for Redis. It first became available in version 1.1. AOF can be enabled in the Redis configuration file:

appendonly yes

Once enabled, every time Redis receives a command that changes the dataset (e.g. SET) it will append it to the AOF. When you restart Redis it will re-play the AOF to rebuild the state. This same file can be used to rebuild / migrate a Redis instance in Azure.

### Manual (SET)

The most basic way to migrate an instance is to enumerate all the keys from the source and then SET the values in the destination. This works well with basic key values such as strings and integers, but care has to be taken with more complex objects such that the tool encodes the values correctly in the migrate process.

### Manual (DUMP/RESTORE)

This path is the preferred path as it will export the key in the Redis encoded format. Although it is the preferred method, it presents various challenges when the source and target are not within a compatible version range for the encoding algorithm.

### SLAVEOF / REPLICAOF

Redis includes the ability to create replicas of master nodes. This path is one of the easiest to setup, but unfortunately none of the Azure services support the SLAVEOF or REPLICAOF commands. This means this path is best used for when you are moving from one version to another to support a move to the cloud using the DUMP and RESTORE path.

### MIGRATE

The [MIGRATE](https://redis.io/commands/migrate) Redis command will atomically transfer a key from a source Redis instance to a destination Redis instance. On success the key is deleted from the original instance and is guaranteed to exist in the target instance.

### Layer of Abstraction

Layer of abstraction means that you can use your applications to migrate your Redis data in real-time and as the data is used. Once you hit 100% key coverage, you can then remove the layer of abstraction and retire the old Redis instances.

### Other open-source tools

There are several 3rd party migration tools that help migrate Redis workloads easily and quickly. In most cases, the time savings and ease of use come with a price and may add extra costs to the migration.

Some of these include:

* [redis-copy](https://github.com/deepakverma/redis-copy)
* [redis-migrate](https://github.com/vipshop/redis-migrate-tool)

## Fastest/Minimum Downtime Migration

As outlined above, there are plenty of paths for migrating cache data. Deciding which path to take is a function of the migration team’s skill set, and the amount of downtime the instance and application owners are willing to accept. Some tools support multi-threaded parallel data migration approaches while other tools were designed for simple migrations of key/value data only.

## Decision Table

There are many paths WWI can take to migrate their Redis workloads. We have provided a table of the potential paths and the advantages and disadvantages of each:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Objective | Description | Tool | Prerequisites | Advantages | Disadvantages |
| Fastest migration possible | Parallel approach | 3rd party tool | Scripted Setup | Highly parallelized | Target throttling |
| Online migration | Keep the source up for as long as possible | Replication | None | Seamless | Extra processing and storage |
| Highly Customized Offline Migration | Selectively export objects | IMPORT/EXPORT | None | Highly customizable | Manual |

## WWI Use Case

WWI has selected its conference instance as its first migration workload. The workload was selected because it had the least risk and the most available downtime due to the gap in the annual conference schedule. They also assessed the instance to not be using any unsupported features in the target Azure Cache for Redis service. Based on the migration team’s other assessment details, they determined that they will attempt to perform an offline migration using the backup and restore Redis tools.

During their assessment period, they did find that the customer instance does use some languages, extensions, and a custom function that are not available in the target service for the conference instance. They have asked the development team to review replacing those features while they migrate the more simple workloads. If they can be replaced successfully, they will choose an Azure Cache for Redis service, otherwise, they will provision an Azure VM to host the workload.

## Migration Methods Checklist

* Ensure the right method is selected given the target and source environments.
* Ensure the method can meet the business requirements.
* Always verify if the data workload will support the method.

# Test Plans

## Overview

WWI created a test plan that included a set of IT and Business tasks. Successful migrations require all the tests to be executed.

Tests:

* Ensure the migrated instance has consistency (same record counts) with on-premises instance.
* Ensure the performance is acceptable (it should match the same performance as if it were running on-premises).
* Ensure acceptable network connectivity between on-premises and the Azure network.
* Ensure all identified applications and users can connect to the migrated data instance.

WWI has identified a migration weekend and time window that started at 10 pm and ended at 2 am Pacific Time. If the migration did not complete before the 2 am target (the 4hr downtime target) with all tests passing, the rollback procedures were started. Issues were documented for future migration attempts. All migrations windows were pushed forward and rescheduled based on acceptable business timelines.

## Sample Queries

A series of queries were executed on the source and target to verify migration success. The following queries and scripts will help determine if the migration moved all required instance objects from the source to the target.

### Exporting Objects

Use this query to get all the keys:

KEYS \*

**NOTE** Running this command on a production environment can cause performance issues with any applications using the target instance. It is advisable to execute during low traffic/usage periods.

Use this query to get all the users:

ACL LIST

TODO : https://redis.io/topics/acl

### Commands

TODO

Command renames (for app level)

### Scripts

If you have any scripts or your application makes calls to the [EVAL command](https://redis.io/commands/eval), be sure to test them thoroughly in case your scripts use commands or features that are not available in Azure Cache for Redis.

## Rollback Strategies

The queries above will provide a list of object names and counts that can be used in a rollback decision. Migration users can take the first object verification step by checking the source and target object counts. A failure in counts may not necessarily mean that a rollback is needed. Performing an in-depth evaluation could point out that the discrepancy is small and easily recoverable. Manual migration of a few failed objects may be possible.

For example, if all keys and values were migrated, but only a few users were missed, remediate those failed items and finalize the migration. If the instance is relatively small, it could be possible to clear the Azure Cache for Redis and restart the migration again. However, if the instance is relatively large, there may not be enough time to determine what occurred. The migration will need to stop and rollback.

Identifying missing instance objects needs to occur quickly during a migration window. Every minute counts. One option could be exporting the environment object names to a file and using a data comparison tool to reduce the missing object verification time. Another option could be exporting the source instance object names and importing the data into a target instance environment temp table. Compare the data using a **scripted** and **tested** query statement. Data verification speed and accuracy are critical to the migration process. Do not rely on reading and verifying a long list of instance objects during a migration window. Manage by exception.

### Decision Table

|  |  |  |  |
| --- | --- | --- | --- |
| Discrepancy | Time To Sync | Rollback? | Resolution Path |
| Key Count Mismatch | Less than the remaining window | No | Sync the missing keys |
| Key Value Mismatch | More than the remaining window | Yes | Start the Rollback |

In the [migration](file:///D:\data\projects\redis-cache\pandoc\03_Migration\01_DataMigration.md) section, we will provide a instance migration inventory script that will provide object counts that can be used to compare source and destination after a migration path has been completed.

## WWI Use Case

The WWI CIO received a confirmation report that all instance objects were migrated from the on-premises instance to the Azure Cache for Redis instance. The infrastructure and dev teams ran the above queries against the instance before the beginning of the migration and saved all the results to a spreadsheet for tracking.

The source instance information was used to verify the target migration object fidelity.

## Checklist

* Have test queries scripted, tested, and ready to execute.
* Know how long test queries take to run and make them a part of the migration timeline.
* Have a mitigation and rollback strategy ready for different potential outcomes.
* Have a well-defined timeline of events for the migration.

# Performance Baselines

Understanding the existing Redis workload is one of the best investments that can be made to ensure a successful migration. Excellent system performance depends on adequate hardware and great application design. Items such as CPU, memory, disk, and networking need to be sized and configured appropriately for the anticipated load. Hardware and configuration are part of the system performance equation. The developer must understand the instance query load and the most expensive queries to execute. Focusing on the most expensive queries can have a big difference in the overall performance metrics.

Creating baselines of query performance is vital to a migration project. The performance baselines can be used to verify the Azure landing zone configuration for the migrated data workloads. Most systems will be run 24/7 and have different peak load times. It is important to capture the peak workloads for the baseline. Later in the document, we will explore the source server parameters and how they are essential to the overall performance baseline picture. The server parameters should not be overlooked during a migration project.

## Tools

Below are tools used to gather server metrics and instance workload information. Use the captured metrics to determine the appropriate Azure Cache for Redis tier and the associated scaling options.

* [redis-benchmark.exe](https://www.percona.com/software/instance-tools/percona-monitoring-and-management): Redis includes the redis-benchmark utility that simulates running commands done by N clients at the same time sending M total queries.

You can find some of the pitfalls and misconceptions of using tools to benchmark Redis [here](https://redis.io/topics/benchmarks).

## redis-benchmark

* Open a terminal to the **SUFFIX-redis01** server
* Run the following:

redis-benchmark -t set -r 100000 -n 1000000

* You should get back something similar to the following:

TODO - Image

## Server Configuration

TODO

## WWI Use Case

TODO

WWI reviewed its Conference instance workload and determined it had a very small load. Although a basic tier server would work for them, they did not want to perform work later to migrate to another tier. The server being deployed will eventually host the other Redis data workloads, so they picked the Premium tier.

In reviewing the Redis instance, the Redis 4.0 server is running with the default server configuration set during the initial install.

# Data Migration with Backup and Restore (RDB)

## Setup

Follow all the steps in the [Setup](file:///D:\data\projects\redis-cache\pandoc\05_Appendix\00_Setup.md) guide to create an environment to support the following steps.

## Disable AOF in Target

To speed up the import, be sure to disable AOF in the target instance.

* Run the following on the target:

redis-cli config set appendonly no

## Data

**NOTE** RDB file format changes between versions may not be backwards compatible.

### Manual Backup

* Run the following command to find where your RDB file is located:

redis-cli config get dir  
  
sudo chown -R $USER /var/lib/redis  
  
chmod 666 /var/lib/redis  
chmod a+x /var/lib/redis  
  
sudo chown redis:redis /var/lib/redis/dump.rdb  
  
sudo chmod 660 /var/lib/redis/dump.rdb

* Create a backup

redis-cli bgsave

* If you have an errors, you can run the following to get error info:

sudo tail /var/log/redis/redis-server.log -n 100

* Install Azure CLI

curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash

* Run the following commands to save your RDB file to azure storage, be sure to replace the subscription id, resource group and storage account tokens:

**NOTE** It can take a couple of minutes for the Azure RBAC assignment to go into effect.

az login  
  
az account set --subscription "<subscription name>"  
  
az ad signed-in-user show --query objectId -o tsv | az role assignment create --role "Storage Blob Data Contributor" --assignee @- --scope "/subscriptions/<subscription>/resourceGroups/<resource-group>/providers/Microsoft.Storage/storageAccounts/<storage-account>"  
  
az storage container create --account-name <storage-account> --name <container> --auth-mode login  
  
az storage blob upload --account-name <storage-account> --container-name redis --name database.rdb --file /var/lib/redis/database.rdb --auth-mode login

### Manual Restore

You can import the data using the Azure Portal or Azure CLI / PowerShell.

**NOTE** If you attempt to import on a Basic or Standard tier, you will get an error that a Premium tier is required.

**NOTE** Each tier supports a maximum number of databases. If you have more than the default of 16, be sure that you pick a tier to migrate too that has support for all source databases.

**NOTE** Azure Cache for Redis supports RDB import up through RDB version 7.

**NOTE** You can import/export between a clustered cache and a non-clustered cache. Since Redis cluster only supports database 0, any data in databases other than 0 isn’t imported. When clustered cache data is imported, the keys are redistributed among the shards of the cluster.

#### Azure Powershell

* Install Azure Powershell

# Update the list of packages  
sudo apt-get update  
# Install pre-requisite packages.  
sudo apt-get install -y wget apt-transport-https software-properties-common  
# Download the Microsoft repository GPG keys  
wget -q https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb  
# Register the Microsoft repository GPG keys  
sudo dpkg -i packages-microsoft-prod.deb  
# Update the list of products  
sudo apt-get update  
# Enable the "universe" repositories  
sudo add-apt-repository universe  
# Install PowerShell  
sudo apt-get install -y powershell

# Start PowerShell  
pwsh  
  
Install-Module -Name Az -AllowClobber  
  
Import-AzRedisCache -ResourceGroupName "resourceGroupName" -Name "cacheName" -Files @("https://mystorageaccount.blob.core.windows.net/mycontainername/blobname?sv=2015-04-05&sr=b&sig=caIwutG2uDa0NZ8mjdNJdgOY8%2F8mhwRuGNdICU%2B0pI4%3D&st=2016-05-27T00%3A00%3A00Z&se=2016-05-28T00%3A00%3A00Z&sp=rwd") -Force

#### Azure Cli

* Perform the import:

az redis import --files "https://mystorageaccount.blob.core.windows.net/mycontainername/blobname?sv=2015-04-05&sr=b&sig=caIwutG2uDa0NZ8mjdNJdgOY8%2F8mhwRuGNdICU%2B0pI4%3D&st=2016-05-27T00%3A00%3A00Z&se=2016-05-28T00%3A00%3A00Z&sp=rwd" --file-format RDB --name "cacheName" --resource-group "resourceGroupName" --subscription "subscriptionName"

#### Azure Portal

* Browse to the Azure Portal
* Select the redis resource group
* Select the target Redis instance (must be Premium or higher)
* Under **Import**, select **Import**
* Select the storage account
* Select the **redis** container
* Select the **dump.rdb** file
* Select **OK**

### Check success

* Depending on the size of the file, it could take a while for the import to finish. Once it is completed, verify that all keys have been migrated.

## Enable AOF in Target

If you disabled AOF in the target, re-enable it:

* Run the following on the target:

redis-cli config set appendonly yes

## Summary - Backup and Restore

Even a simple backup and restore operation can potentially require significant effort to restore to an Azure Cache for Redis instance.

Practice the above steps and record the time it takes to complete the entire migration.

## Resources

* [Import and Export data in Azure Cache for Redis](https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-how-to-import-export-data) # Data Migration with Redis (Mass Insertion)

## Setup

Follow all the steps in the [Setup](file:///D:\data\projects\redis-cache\pandoc\05_Appendix\00_Setup.md) guide to create an environment to support the following steps.

## Connectivity Setup

In order to connect to Azure Redis, you must either enable non-SSL port, or you must install a tunneling tool such as [STunnel](file:///D:\data\projects\redis-cache\pandoc\05_Appendix\03_InstallStunnel.md).

### Enable Non-SSL Port

* Open the Azure Portal
* Browse to the Azure Cache for Redis
* Select **Non–SSL port (6379) disabled** link
* Select to enable the port
* Select **Save**

## Disable AOF in Target

To speed up the import, be sure to disable AOF in the target instance.

* Run the following on the target:

redis-cli config set appendonly no

## Mass Insertion (SET)

* Run the following command to export all keys and values to a file
* Start a PowerShell session:

pwsh

* Run the following to dump all the keys and values:

del "set.txt";  
  
$keys = redis-cli keys "\*"  
  
foreach($key in $keys)  
{  
 $val = redis-cli get $key  
  
 $line = $key + "`t" + $val;  
 add-content "set.txt" $line;  
}

* Run the following command to import all keys and values

$lines = get-content "set.txt";  
  
foreach($line in $lines)  
{  
 $vals = $line.split("`t");  
  
 $key = $vals[0];  
 $val = $vals[1];  
  
 $ret = $val | redis-cli -h <redis name>.redis.cache.windows.net set $key $val;  
  
 Write-Host "Setting $key to $val";  
}

**NOTE** This approach will lose important metadata such as Time To Live (TTL) settings that may be set in the source. You should also be sure that all values that are export and then re-imported are encoded in a way that ensures valid values imported into the target instance.

## Mass Insertion (DUMP/RESTORE)

export OLD="redis-cli -h localhost"  
export NEW="redis-cli -h <REDIS\_NAME>.redis.cache.windows.net -a <REDIS\_KEY>"  
  
  
for KEY in $($OLD --scan); do  
 $OLD --raw DUMP "$KEY" | head -c-1 > /tmp/dump  
 TTL=$($OLD --raw TTL "$KEY")  
 case $TTL in  
 -2)  
 $NEW DEL "$KEY"  
 ;;  
 -1)  
 $NEW DEL "$KEY"  
 cat /tmp/dump | $NEW -x RESTORE "$KEY" 0  
 ;;  
 \*)  
 $NEW DEL "$KEY"  
 cat /tmp/dump | $NEW -x RESTORE "$KEY" "$TTL"  
 ;;  
 esac  
 echo "$KEY (TTL = $TTL)"  
done

**NOTE** Add the -c option if running against a cluster as the source or target.

## Mass Insertion (MIGRATE)

You can use the MIGRATE command to move key values to a new instance.

* If using stunnel, run the following:

redis-cli --scan | xargs redis-cli MIGRATE localhost 6380 "" 0 15000 AUTH <REDIS\_PWD> KEYS

If using the non-SSL port, run the following:

redis-cli --scan | xargs redis-cli MIGRATE <REDIS\_NAME>.redis.cache.windows.net 6  
379 "" 0 15000 AUTH <REDIS\_KEY> KEYS

**NOTE** The target versions must be the same, or have similar encoding of the values in order for the MIGRATE command to succeed.

## Enable AOF in Target

If you disabled AOF in the target, re-enable it:

* Run the following on the target:

redis-cli config set appendonly yes

# Data Migration with Redis Replication

Using the SLAVEOF or the MIGRATEOF is for migrating from one version to another to support a move to Azure.

**NOTE** None of the Azure services (basic, standard, premimum, or enterprise support the SLAVEOF or REPLICAOF commands).

Taking this path is an intermediate step for when a migration tool has been selected that uses the DUMP and RESTORE commands. These commands will not work if the two instances are different because the encoding of the key value does not port properly.

## Setup

Follow all the steps in the [Setup](file:///D:\data\projects\redis-cache\pandoc\05_Appendix\00_Setup.md) guide to create an environment to support the following steps.

**Note** If the server is not enabled for SSL and is running 6.0 or higher, it is highly recommended that it is configured. See [Configure Redis for SSL connectivity](file:///D:\data\projects\redis-cache\pandoc\05_Appendix\04_ConfigureRedisSSL.md) to enable SSL for the instance.

## Option 1 : Set to Read Only

* Run the following to set the source Redis server to read only:

redis-cli config set min-replicas-to-write 3  
  
sudo service redis-server restart

## Option 2 : Rename Commands

This can also be accomplished by rename-ing all write commands.

* Open the redis.conf file, add the following lines:

rename-command FLUSHDB ""  
rename-command FLUSHALL ""  
rename-command DEBUG ""  
rename-command SET ""  
rename-command DEL ""  
rename-command RENAME ""  
rename-command RPUSH ""  
rename-command SADD ""  
rename-command ZADD ""  
rename-command HMSET ""  
rename-command EXPIRE ""  
  
sudo service redis-server restart

**NOTE** This is not all write commands, add what you need to prevent any changes to your Redis instance during the migration.

## Replication (SLAVEOF) : Pre Redis 5.0

* Browse to the Azure Portal
* Select the **SUFFIX-redis01** Redis instance
* Open a PowerShell window, login using SSH
* Run the following commands:

redis-cli

## Replication (REPLICAOF) : 5.x or higher

* Browse to the Azure Portal
* Select the **SUFFIX-redis02** Redis instance
* Open a PowerShell window, login using SSH
* Run the following commands:

redis-cli  
  
replicaof source\_hostname\_or\_ip source\_port

* Once replication has completed, run the following on the target:

replicaof no one

## Resources

* [SLAVEOF](https://redis.io/commands/SLAVEOF)
* [REPLICAOF](https://redis.io/commands/REPLICAOF) # Data Migration with 3rd Party Tools

## Setup

Follow all the steps in the [Setup](file:///D:\data\projects\redis-cache\pandoc\05_Appendix\00_Setup.md) guide to create an environment to support the following steps.

**Note** If the source server is not enabled for SSL and is running 6.0 or higher, it is highly recommended that it is enabled and configured. See [Configure Redis for SSL connectivity](file:///D:\data\projects\redis-cache\pandoc\05_Appendix\04_ConfigureRedisSSL.md) to enable SSL for the instance.

**NOTE** Tools that use the REPLICAOF commands will not work with Azure Cache for Redis instances.

## Install STunnel

If you have not done so already, setup [STunnel](file:///D:\data\projects\redis-cache\pandoc\05_Appendix\03_InstallStunnel.md) if you are not running redis-cli 6.x or higher.

## Disable AOF in Target

To speed up the import, be sure to disable AOF in the target instance.

* Run the following on the target:

redis-cli config set appendonly no

## Redis-migrate

* https://github.com/vipshop/redis-migrate-tool
* Download the source and compile the tool by running the following:

sudo apt-get install git-all  
sudo apt-get install automake libtool autoconf bzip2  
  
git clone https://github.com/vipshop/redis-migrate-tool  
  
cd redis-migrate-tool  
autoreconf -fvi  
./configure  
make

* Create a migration configuration file call migrate.conf
* Run the following:

sudo nano migrate.conf

* Copy the following into it, be sure to replace the target REdis instance name, port and password:

[source]  
type: single  
servers:  
 - 127.0.0.1:6379  
  
[target]  
type: single  
redis\_auth: <REDIS\_PWD>  
servers:  
 - <REDIS\_NAME>.redis.cache.windows.net:6379:1 server1  
  
[common]  
listen: 0.0.0.0:8888  
threads: 2  
step: 1  
mbuf\_size: 1024  
source\_safe: true

* To use the tool, run the following:

./src/redis-migrate-tool -c migrate.conf -o log -d

TODO - does this work?

* Check the status of the tool:

redis-cli -h 127.0.0.1 -p 8888

* Check that data in the source and target:

./src/redis-migrate-tool -c migrate.conf -o log -C redis\_check

* Run a test insert

./src/redis-migrate-tool -c migrate.conf -o log -C "redis\_testinsert"

## redis-copy (yaauie)

TODO

https://github.com/yaauie/redis-copy

## redis-copy (deepakverma)

This is a .NET based tool that uses the StackExchange.Redis library to move keys from the source to the destination. It uses the DUMP and RESTORE commands to make the move, which means the source and the destination should support the same encoding of the DUMP value. Although the tools has not had any changes made to it in quite a while, the source is a great starting point for building your own version.

* Switch to the **win10** virtual machine
* Open a PowerShell window, run the following:

git clone https://github.com/deepakverma/redis-copy

* Open the redis-copy.sln solution file
* Right-click the project file, select **Manage Nuget Packages**
* Select the **Installed** tab, then select **StackExchange.Redis**
* Update the package to be the latest version
* Press **Ctrl-B** to build the solution
* If the source system is running SSL, run the following,

cd /redis-copy/bin/debug  
  
redis-copy.exe --se localhost --sa "" --sp 6380 --sssl --db 0 --de <REDIS\_NAME>.redis.cache.windows.net --da <REDIS\_PWD> --dp 6380 --dssl --flushdest

* If the source system is not running SSL, run the following:

cd /redis-copy/bin/debug  
  
redis-copy.exe --se localhost --sa "" --sp 6379 --db 0 --de <REDIS\_NAME>.redis.cache.windows.net --da <REDIS\_PWD> --dp 6380 --dssl --flushdest

**NOTE** You may need to make changes to the source to handle error such as DUMP/RESTORE encoding issues, this can be done by adding a method to check the version of the source and target Redis instances before you start the key migration.

## redis-dump

https://www.npmjs.com/package/redis-dump

## redis-dump

Redis-dump is a ruby based tool that also uses the DUMP and RESTORE commands to migrate from a source to a destination instance.

* On the target server, run the following to install the tool:

sudo apt-get install ruby ruby-dev  
  
gem install redis-dump -V

* Run the following to run the tool:

TODO

## Enable AOF in Target

If you disabled AOF in the target, re-enable it:

* Run the following on the target:

redis-cli config set appendonly yes

## Resources

* https://github.com/vipshop/redis-migrate-tool
* https://github.com/deepakverma/redis-copy # Data Migration with Layer of Abstraction

If migrating an entire Redis instance is not feasible or worth the extra effort, you can add a layer of abstraction in your application code that can be enabled or disabled via configuration.

This extra layer of code will be responsible for retrieving the cache value from the source instance, and adding it to the destination instance. You can set a threshold for the number of key coverage (such as 90%) you are targeting and once that is hit, you can change the application to start pulling values from the new instance.

## Example

In the following .NET code, you can modify your application to add a method that still returns the cache value, but also does the extra work to save the value to the new instance.

TODO

**NOTE** It is important to ensure that the save operation to the new instance is async to prevent any application performance issues.

# Data Migration

## Back up the instance

Lower risk and back up the instance before upgrading or migrating data. Use the Redis save or bgsave command to backup the Redis data to disk.

## Offline vs. Online

Before selecting a migration tool, decide if the migration should be online or offline.

* **Offline migrations** require the system to be down while the migration takes place. Users will not be able to modify data. This option ensures that the state of the data will be exactly what is expected when restored in Azure.
* **Online migrations** will migrate the data in near real-time. This option is appropriate when there is little downtime for the users or application consuming the data workload. The costs are too high for the corporation to wait for complete migration. The process involves replicating the data using some type of replication method.

**Case Study:** In the case of WWI, their environment has some complex networking and security requirements that will not allow for the appropriate changes to be applied for inbound and outbound connectivity in the target migration time frame. These complexities and requirements essentially eliminate the online approach from consideration.

**Note:** Review the Planning and Assessment sections for more details on Offline vs Online migration.

## Data Drift

Offline migration strategies have the potential for data drift. Data drift occurs when newly modified source data becomes out of sync with migrated data. When this happens, a full export or a delta export is necessary. To mitigate this problem, stop all write traffic to the instance and then perform the export. If stopping all data modification traffic is not possible, it will be necessary to account for the data drift as part of the migration effort.

Determining the changes can be complicated if you do not have a tracking mechanism. Luckily, Redis has the AppendOnly feature that will generate a log file of all key changes. This could be used as the diff of the instance from a particular point (such as the start of the migration).

## Performance recommendations

### Source Tool Network

When running the migration tool on a virtual machine, it is possible to change the TCP\_NODELAY setting. By default, TCP uses Nagle’s algorithm, which optimizes by batching up outgoing packets. This means fewer sends and this works well if the application sends packets frequently and latency is not the highest priority. Realize latency improvements by sending on sockets created with the TCP\_NODELAY option enabled. This results in lower latency but more sends. Consider this client-side setting for the Virtual Machine (VM). Applications that benefit from the TCP\_NODELAY option typically tend to do smaller, infrequent writes and are particularly sensitive to latency. As an example, alter this setting to reduce latency from 15-40 ms to 2-3 ms.

To change this setting on Windows machines, do the following:

* Open the REGEDIT tool
* Under the subtree HKEY\_LOCAL\_MACHINE, find the SYSTEM\CurrentControlSet\services\Tcpip\Parameters\Interfaces key
* Find the correct network interface
* In the space, right-click and select **New** for creating a DWORD Value
* For the value name, type **TcpNoDelay**
* For the Dword value, type **1**
  + In the empty space right-click, and select **New** for creating a DWORD Value
* For the value name, type **TcpAckFrequency**
* For the Dword value, type **1**
* Close the REGEDIT tool

### Exporting

TODO - Use an export tool that leverages multiple threads.

### Importing

When you select a path for migration, you will want the import to run as fast as possible. You can disable the AOF feature to get faster imports, then re-enable it when you are done with the migration.

## Performing the Migration

* Back up the instance
* Create and verify the Azure Landing zone
* Export and configure Source Server configuration
* Export and configure Target Server configuration
* Export the instance objects (schema, users, etc.)
* Export the data
* Import the instance objects
* Import the data (no triggers, keys)
* Validation
* Migrate the Application(s)

## Common Steps

Despite what path is taken, there are common steps in the process:

* Upgrade to a supported Azure Redis version that matches the target
* Inventory instance objects
* Export users and permissions (ACLS)
* Export and configuration settings

### Post Import

* Setup Compliance and Security features
* Configure monitoring of the instance

## WWI Use Case

TODO

## instance Objects

As outlined in the [Test Plans](file:///D:\data\projects\redis-cache\pandoc\02_PreMigration\04_TestPlans.md) section, take an inventory of instance objects before and after the migration.

Migration teams should develop and test helpful inventory scripts before beginning the migration phase.

Instance object inventory examples:

TODO

## Execute migration

With the basic migration components in place, it is now possible to proceed with the data migration. WWI will utilize the Redis pgAdmin option to export the data and then import it into Azure Cache for Redis.

Options:

* [Backup and Restore](file:///D:\data\projects\redis-cache\pandoc\temp\01.01_DataMigration_BackupRestore.md)
* [Copy command](file:///D:\data\projects\redis-cache\pandoc\temp\01.02_DataMigration_Copy.md)
* [Replication](file:///D:\data\projects\redis-cache\pandoc\temp\01.03_DataMigration_Replication.md)
* [3rd Party Tools](file:///D:\data\projects\redis-cache\pandoc\temp\01.04_DataMigration_Tools.md)
* [Layer of abstraction](file:///D:\data\projects\redis-cache\pandoc\temp\01.05_DataMigration_Abstraction.md)

Once the data is migrated, point the application to the new instance

* [Migrate Application Settings](file:///D:\data\projects\redis-cache\pandoc\temp\02_DataMigration_AppSettings.md)

Lastly, validate the target instance’s inventory. Below is an example of the INFO results in a target environment. It is relatively easy to identify database key count discrepancies.

TODO IMAGE

## Data Migration Checklist

* Understand the complexity of the environment and determine if an online approach is feasible.
* Account for data drift. Stopping the instance service can eliminate potential data drift. Acceptable downtime costs?
* Configure source parameters for fast export.
* Configure target parameters for fast import.
* Test any migrations that have a different source version vs the target.
* Migrate any miscellaneous objects, such as user names and privileges.
* Update application settings to point to the new instance.
* Document all tasks. Update task completion to indicate progress.

# Data Migration - Configuration

## Configuration

It is possible that you may want to keep your source instance configuration settings and move them over to Azure Cache for Redis.

### Export the configuration

* Run the following to export any valuable configuration details

TODO

* Items to consider
  + Persistence type
  + Cache Eviction policy

### Export Users and Roles (6.x or higher)

If you are using Redis version 6.x or higher, you may have created ACL rules that will need to be migrated.

* Run the following to export all users and ACLs

$users = redis-cli ACL LIST  
  
foreach($user in $users)  
{  
 $vals = $user.split(" ");  
  
 $user = $vals[1];  
  
 foreach($val in $vals)  
 {  
 if ($val.startswith("#"))  
 {  
 $hashPwd = $val;  
 }  
  
 if ($val.startswith("~"))  
 {  
 $keys = $val;  
 }  
  
 if ($val.startswith("&"))  
 {  
 $channels = $val;  
 }  
  
 if ($val.startswith("+"))  
 {  
 $commands = $val;  
 }  
 }  
  
 $line = $user + "`t" + $keys + "`t" + $commands + "`t" + $hashPwd + "`t"  
  
 add-content "users.txt" $user;  
}

* Import into the target instance

$lines = get-content "users.txt"  
  
  
foreach($line in $lines)  
{  
 $vals = $line.split("`t");  
  
 #parse the values  
 $user = $vals[0];  
 $hashPwd = $vals[3];  
 $keys = $vals[1];  
 $commands = $vals[2];  
  
 redis-cli ACL SETUSER $user on $keys +$commands `#$hashPwd  
}

After running through the previous series of steps, the migration of the source configuration to the target will be completed.

# Data Migration - Application Settings

Most applications use Redis client libraries to handle communication with their caches. In some cases, you may need to upgrade the client library to get the SSL supported version. Once you have that version, you may also need to make code changes to support SSL. Reference [Configure your application to use TLS 1.2](https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-remove-tls-10-11#configure-your-application-to-use-tls-12) for more information.

## Setup

Follow all the steps in the [Setup](file:///D:\data\projects\redis-cache\pandoc\05_Appendix\00_Setup.md) guide to create an environment to support the following steps.

## Update Applications to support SSL

* Open Redis-Testing project
* Update the following code:

TODO

* Update the Connectionstring

TODO

## Redeploy the application

* TODO

## Test the application

* TODO

You have successfully completed an on-premises to Azure Cache for Redis migration!

# Post Migration Management

## Monitoring and Alerts

Once the migration has been successfully completed, the next phase it to manage the new cloud-based cache workload resources. Management operations include both control plane and data plane activities. Control plane activities are those related to the Azure resources versus data plane which is **inside** the Azure resource (in this case Redis).

Azure Cache for Redis provides for the ability to monitor both of these types of operational activities using Azure-based tools such as [Azure Monitor](https://docs.microsoft.com/en-us/azure/azure-monitor/overview), [Log Analytics](https://docs.microsoft.com/en-us/azure/azure-monitor/platform/design-logs-deployment) and [Azure Sentinel](https://docs.microsoft.com/en-us/azure/sentinel/overview). In addition to the Azure-based tools, security information and event management (SIEM) systems can be configured to consume these logs as well.

Whichever tool is used to monitor the new cloud-based workloads, alerts will need to be created to warn Azure and instance administrators of any suspicious activity. If a particular alert event has a well-defined remediation path, alerts can fire automated [Azure run books](https://docs.microsoft.com/en-us/azure/automation/automation-quickstart-create-runbook) to address the event.

The first step to creating a fully monitored environment is to enable Redis log data to flow into Azure Monitor. Reference [monitor Azure Cache for Redis](https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-how-to-monitor) for more information.

Once log data is flowing, use the [Kusto Query Language (KQL)](https://docs.microsoft.com/en-us/azure/data-explorer/kusto/query/) query language to query the various log information. Administrators unfamiliar with KQL can find a SQL to KQL cheat sheet [here](https://docs.microsoft.com/en-us/azure/data-explorer/kusto/query/sqlcheatsheet) or the [Get started with log queries in Azure Monitor](https://docs.microsoft.com/en-us/azure/azure-monitor/log-query/get-started-queries) page.

For example, to get the memory usage of the Azure Cache for Redis:

AzureMetrics  
| where TimeGenerated > ago(15m)  
| where ResourceProvider == "MICROSOFT.CACHE"  
| where MetricName == "usedmemory"  
| limit 10  
| project TimeGenerated, Total, Maximum, Minimum, TimeGrain, UnitName  
| top 1 by TimeGenerated

To get the CPU usage:

AzureMetrics  
| where TimeGenerated > ago(15m)  
| where ResourceProvider == "MICROSOFT.CACHE"  
| where MetricName == "allpercentprocessortime"  
| limit 10  
| project TimeGenerated, Total, Maximum, Minimum, TimeGrain, UnitName  
| top 1 by TimeGenerated

**Note** for a list of other metrics, reference [Monitor Azure Cache for Redis](https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-how-to-monitor).

Once a KQL query has been created, create [log alerts](https://docs.microsoft.com/en-us/azure/azure-monitor/platform/alerts-unified-log) based off these queries.

## Server Configuration

As part of the migration, it is likely the on-premises server configuration was modified to support a fast egress. Also, modifications were made to the Azure Cache for Redis configuration to support a fast ingress. The Azure Redis instance configuration should be set back to their original on-premises workload optimized values after the migration.

However, be sure to review and make server configuration changes that are appropriate for the workload and the environment. Some values that were great for an on-premises environment, may not be optimal for a cloud-based environment. Additionally, when planning to migrate the current on-premises configuration to Azure, verify that they can in fact be set.

## PowerShell Module

The Azure Portal and Windows PowerShell can be used for managing the Azure Cache for Redis. To get started with PowerShell, install the Azure PowerShell cmdlets for Redis with the following PowerShell command:

Install-Module -Name Az.RedisCache

After the modules are installed, reference tutorials and documentation like the following to learn ways to take advantage of scripting various management activities:

* [Manage Azure Cache for Redis with Azure Powershell](https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-how-to-manage-redis-cache-powershell)

## Azure Cache for Redis Upgrade Process

Since Azure Cache for Redis is a PaaS offering, administrators are not responsible for the management of the updates on the operating system or the Redis software. However, it is important to be aware the upgrade process can be random and when being deployed, will stop the Redis server workloads. Plan for these downtimes by rerouting the workloads to a replica in the event the particular instance goes into maintenance mode, or utilize a replication based SKU to ensure uptime.

**Note:** This style of failover architecture may require changes to the applications layer to support this type of failover scenario. If the replica is maintained as a read only replica and is not promoted, the application will only be able to read data and it may fail when any operation attempts to write information to the instance.

The [AzureRedisEvents](https://docs.microsoft.com/en-us/azure/Redis/concepts-planned-maintenance-notification) feature will inform resource owners up to 30 seconds in advance of installation of an update or critical security patch. Instance administrators may need to notify application users of planned and unplanned maintenance.

**Note:** Azure Cache for Redis maintenance notifications are incredibly important. The instance maintenance can take the instance and connected applications down for a random period of time. Nodes are patched one at a time to prevent data loss. Basic caches will have data loss. Clustered caches are patched one shard at a time.

## WWI Use Case

WWI decided to utilize the Azure Activity logs and enable Redis logging to flow to a [Log Analytics workspace](https://docs.microsoft.com/en-us/azure/azure-monitor/platform/design-logs-deployment). This workspace is configured to be a part of [Azure Sentinel](https://docs.microsoft.com/en-us/azure/sentinel/) such that any [Threat Analytics](https://docs.microsoft.com/en-us/azure/Redis/concepts-data-access-and-security-threat-protection) events would be surfaced, and incidents created.

## Management Checklist

* Create resource alerts for common things like CPU and Memory.
* Ensure the server is configured for the target data workload after migration.
* Script common administrative tasks.
* Set up notifications for maintenance events such as upgrades and patches. Notify users as necessary.
* Ensure the application can handle maintenance activities

# Optimization

## Monitoring Hardware and Cache Performance

In addition to the audit and activity logs, cache performance can also be monitored with [Azure Metrics](https://docs.microsoft.com/en-us/azure/azure-monitor/platform/data-platform-metrics). Azure metrics are provided in a one-minute frequency and alerts can be configured from them. For more information, reference [Monitoring in Azure Cache for Redis](https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-how-to-monitor) for specifics on what kind of metrics that can be monitored.

As previously mentioned, monitoring metrics such as the allpercentprocessortime or usedmemory can be important when deciding to upgrade the instance tier. Consistently high values could indicate a tier upgrade is necessary.

Additionally, if cpu and memory do not seem to be the issue, administrators can explore instance-based options such as cache misses.

To find cache misses, run the following:

AzureMetrics  
| where ResourceProvider == "MICROSOFT.CACHE"  
| where MetricName == 'cachemisses'  
| limit 10  
| project TimeGenerated, Total, Maximum, Minimum, TimeGrain, UnitName  
| top 1 by TimeGenerated

## Typical Workloads and Performance Issues

There tends to be two common usage patterns with any instance system, Redis included. These are (but are not limited too):

* An application server exposing a web endpoint on an application server, which connects to the instance.
* A client-server architecture where the client directly connects to the instance.

In consideration of the above patterns, performance issues can crop up in any of the following areas:

* **Resource Contention (Client)** - The machine/server serving as the client could be having a resource constraint which can be identified in the task manager, the Azure portal, or CLI if the client machine is running on Azure.
* **Resource Contention (Application Server)** - The machine/server acting as the application server could cause a resource constraint, which can be identified in the task manager, the Azure portal, or CLI if the application server/service VM is running on Azure. If the application server is an Azure service or virtual machine, then Azure metrics can help with determining the resource contention.
* **Resource Contention (instance Server)** - The instance service could be experiencing performance bottlenecks related to CPU, memory, and storage which can be determined from the Azure Metrics for the instance service instance.
* **Network latency** - A quick check before starting any performance benchmarking run is to determine the network latency between the client and instance using a simple SELECT 1 query. In most Azure regions, watch for less than two milliseconds of latency on SELECT 1 timing when using a remote client hosted on Azure in the same region as the Azure Cache for Redis server.

## Azure Monitor

[Azure Monitor for Azure Cache for Redis](https://docs.microsoft.com/en-us/azure/azure-monitor/insights/redis-cache-insights-overview) will provide you with common metrics reporting that are specific to Azure Cache for Redis.

## Scale the Server

It is possible to [scale up the tier](https://docs.microsoft.com/en-us/azure/azure-cache-for-redis/cache-how-to-scale) at any time, however, once scaled up, you cannot scale the instance down. You would need to re-create a lower tiered instance and then migrate to it.

## Moving Regions

Moving a instance to a different Azure region depends on the approach and architecture. Depending on the selected approach, it could cause system downtime.

The recommended process is the same as utilizing cluster replicas for maintenance failover. However, compared to the planned maintenance method mentioned above, the speed to failover is much faster when a failover layer has been implemented in the application. The application should only be down for a few moments during the read replica failover process. More details are covered in the [Business Continuity and Disaster Recovery](file:///D:\data\projects\redis-cache\pandoc\temp\03_BCDR.md) section.

## Quick Tips

Use the following to make quick performance changes:

* **Server load** : If your instance can’t handle the request, you may need to scale up or add more shards to the cluster.
* **CPU Usage** : If CPU usage for an Azure Cache for Redis server is saturated at 100%, then select the next higher level of Compute Units to get more CPU.
* **Memory Usage** : If memory usage for an Azure Cache for Redis server is saturated at 100%, then select the next higher level of tier to get more memory.
* **Network** : If the tier runs out of bandwidth to serve the clients, the clients will start received time outs, scale up to a higher tier to get more bandwidth.
* **Regions** : It is recommended having the application server/client machine in the same region in Azure to reduce latency between the client/application server and the instance.

## WWI Use Case

WWI business and application users expressed a high level of excitement regarding the ability to scale the instance on-demand.

They opted to utilize a read replica server for any potential failover or read-only needed scenarios.

The migration team, working with the Azure engineers, set up KQL queries to monitor for any potential issues with the Redis server performance. The KQY queries were set up with alerts to email event issues to the instance and conference team.

They elected to monitor any potential issues for now and implement Azure Automation run books at a later time, if needed, to improve operational efficiency.

## Optimization Checklist

* Periodically review the Performance and Azure Advisor recommendations.
* Utilize monitoring to drive tier upgrades and scale decisions.
* Consider moving regions if the users or application needs change.

# Business Continuity and Disaster Recovery (BCDR)

## Backup and Restore

As with any mission critical system, having a backup and restore as well as a disaster recovery (BCDR) strategy is an important part of the overall system design. If an unforseen event occurs, it is important to have the ability to restore the data to a point in time (Recovery Point Objective) and in a reasonable amount of time (Recovery Time Objective).

### Backup

Azure Cache for Redis supports automatic backups for 7 days by default. It may be appropriate to modify this to the current maximum of 35 days. It is important to be aware that if the value is changed to 35 days, there will be charges for any extra backup storage over 1x of the storage allocated.

There are several limitations to the instance backup features as described in each of the backup articles for each service type. It is important to understand them when deciding what additional strategies that should be implemented:

TODO

* [Backup and restore in Azure Cache for Redis](https://docs.microsoft.com/en-us/azure/Redis/concepts-backup)

Commonality of the backup architectures include:

* Up to 35 days of backup protection
* No direct access to the backups (no exports).

Some items to be aware of include:

* Tiers that allow up to 4TB will retain two full backups, all diff backups and transaction logs since last full backup every 7 days.
* Tiers that allow up to 16TB will retain the full backup, all diff backups, and transaction logs in the last 8 days.

**Note:** [Some regions](https://docs.microsoft.com/en-us/azure/Redis/concepts-pricing-tiers#storage) do not yet support storage up to 16TB.

### Restore

Redundancy (local or geo) must be configured during server creation. However, a geo-restore can be performed and allows the modification of these options during the restore process. Performing a restore operation will temporarily stop connectivity and any applications will be down during the restore process.

During a instance restore, any supporting items outside of the instance will also need to be restored. See [Perform post-restore tasks](https://docs.microsoft.com/en-us/azure/Redis/concepts-backup#perform-post-restore-tasks) for more information.

## Replicas

### Read Replicas

[Read replicas](https://docs.microsoft.com/en-us/azure/Redis/concepts-read-replicas) can be used to increase the Redis read throughput, improve performance for regional users and to implement disaster recovery. When creating one or more read replicas, be aware that additional charges will apply for the same compute and storage as the primary server.

## Deleted Servers

If an administrator or bad actor deletes the server in the Azure Portal or via automated methods, all backups and read replicas will also be deleted. It is important that [resource locks](https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/lock-resources) are created on the Azure Cache for Redis resource group to add an extra layer of deletion prevention to the instances.

## Regional Failure

Although rare, if a regional failure occurs geo-redundant backups or a read replica can be used to get the data workloads running again. It is best to have both geo-replication and a read replica available for the best protection against unexpected regional failures.

**Note** Changing the instance server region also means the endpoint will change and application configurations will need to be updated accordingly.

### Load Balancers

If the application is made up of many different instances around the world, it may not be feasible to update all of the clients. Utilize an [Azure Load Balancer](https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-overview) or [Application Gateway](https://docs.microsoft.com/en-us/azure/application-gateway/overview) to implement a seamless failover functionality. Although helpful and time-saving, these tools are not required for regional failover capability.

## WWI Use Case

WWI wanted to test the failover capabilities of read replicas so they performed the steps outlined below.

### Creating a read replica

* Open the Azure Portal.
* Browse to the Azure Cache for Redis instance.
* Under **Settings**, select **Replication**.
* Select **Add Replica**.
* Type a server name.
* Select the region.
* Select **OK**, wait for the instance to deploy. Depending on the size of the main instance, it could take some time to replicate.

**Note:** Each replica will incur additional charges equal to the main instance.

### Failover to read replica

Once a read replica has been created and has completed the replication process, it can be used for failed over. Replication will stop during a failover and make the read replica its own main instance.

Failover Steps:

* Open the Azure Portal.
* Browse to the Azure Cache for Redis instance.
* Under **Settings**, select **Replication**.
* Select one of the read replicas.
* Select **Stop Replication**. This will break the read replica.
* Modify all applications connection strings to point to the new main instance.

## BCDR Checklist

* Modify backup frequency to meet requirements.
* Setup read replicas for read intensive workloads and regional failover.
* Create resource locks on resource groups.
* Implement a load balancing strategy for applications for quick failover.

# Security

Moving to a cloud-based service doesn’t mean the entire internet will have access to it at all times. Azure provides best in class security that ensures data workloads are continually protected from bad actors and rouge programs.

## Authentication

Azure Cache for Redis supports the basic authentication mechanisms for Redis user connectivity, but also supports [integration with Azure Active Directory](https://docs.microsoft.com/en-us/azure/Redis/concepts-aad-authentication). This security integration works by issuing tokens that act like passwords during the Redis login process. [Configuring Active Directory integration](https://docs.microsoft.com/en-us/azure/Redis/howto-configure-sign-in-azure-ad-authentication) is incredibly simple to do and supports not only users, but AAD groups as well.

This tight integration allows administrators and applications to take advantage of the enhanced security features of [Azure Identity Protection](https://docs.microsoft.com/en-us/azure/active-directory/identity-protection/overview-identity-protection) to further surface any identity issues.

**Note:** Be sure to test the application with Azure AD Authentication. See [Use Azure Active Directory for authentication with Redis](https://docs.microsoft.com/en-us/azure/Redis/howto-configure-sign-in-aad-authentication) for more information.

## Threat Protection

In the event that user or application credentials are compromised, logs are not likely to reflect any failed login attempts. Compromised credentials can allow bad actors to access and download the data. [Azure Threat Protection](https://docs.microsoft.com/en-us/azure/Redis/howto-instance-threat-protection-portal) can watch for anomalies in logins (such as unusual locations, rare users or brute force attacks) and other suspicious activities. Administrators can be notified in the event something does not look right.

## Audit Logging

Redis has a robust built-in logging feature. By default, this [log feature is disabled](https://docs.microsoft.com/en-us/azure/Redis/concepts-server-logs) in Azure Cache for Redis. Server level logging can be enabled or modified by changing various server parameters. Once enabled, logs can be accessed through [Azure Monitor](https://docs.microsoft.com/en-us/azure/azure-monitor/overview) and [Log Analytics](https://docs.microsoft.com/en-us/azure/azure-monitor/platform/design-logs-deployment) by turning on [diagnostic logging](https://docs.microsoft.com/en-us/azure/Redis/howto-configure-audit-logs-portal#set-up-diagnostic-logs).

To query for log related events, run the following KQL query:

AzureDiagnostics  
| where LogicalServerName\_s == "myservername"  
| where Category == "RedisLogs"  
| where TimeGenerated > ago(1d)

In addition to the basic logging feature, gain access to more in-depth [audit logging information](https://docs.microsoft.com/en-us/azure/Redis/concepts-audit) that is provided with the pgaudit extension.

AzureDiagnostics  
| where LogicalServerName\_s == "myservername"  
| where TimeGenerated > ago(1d)   
| where Message contains "AUDIT:"

## Encryption

Data in the Redis instance is encrypted at rest by default. Any automated backups are also encrypted to prevent potential leakage of data to unauthorized parties. This encryption is typically performed with a key that is created when the instance is created. In addition to this default encryption key, administrators have the option to [bring your own key (BYOK)](https://docs.microsoft.com/en-us/azure/Redis/concepts-data-encryption-Redis).

When using a customer-managed key strategy, it is vital to understand responsibilities around key lifecycle management. Customer keys are stored in an [Azure Key Vault](https://docs.microsoft.com/en-us/azure/key-vault/general/basic-concepts) and then accessed via policies. It is vital to follow all recommendations for key management as the loss of the encryption key equates to the loss of data access.

In addition to a customer-managed keys, use service-level keys to [add double encryption](https://docs.microsoft.com/en-us/azure/Redis/concepts-infrastructure-double-encryption). Implementing this feature will provide highly encrypted data at rest, but it does come with encryption performance penalties. Testing should be performed.

Data can be encrypted during transit using SSL/TLS. As previously discussed, it may be necessary to [modify your applications](https://docs.microsoft.com/en-us/azure/Redis/concepts-ssl-connection-security) to support this change and also configure the appropriate TLS validation settings.

## Firewall

Once users are set up and the data is encrypted at rest, the migration team should review the network data flows. Azure Cache for Redis provides several mechanisms to secure the networking layers by limiting access to only authorized users, applications and devices.

The first line of defense for protecting the Redis instance is to implement [firewall rules](https://docs.microsoft.com/en-us/azure/Redis/concepts-firewall-rules). IP addresses can be limited to only valid locations when accessing the instance via internal or external IPs. If the Redis instance is destined to only serve internal applications, then [restrict public access](https://docs.microsoft.com/en-us/azure/Redis/concepts-data-access-and-security-private-link#deny-public-access-for-azure-instance-for-Redis-single-server).

When moving an application to Azure along with the Redis workload, it is likely there will be multiple virtual networks setup in a hub and spoke pattern that will require [Virtual Network Peering](https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-peering-overview) to be configured.

## Private Link

To limit access to the Azure Cache for Redis to internal Azure resources, enable [Private Link](https://docs.microsoft.com/en-us/azure/Redis/concepts-data-access-and-security-private-link). Private Link will ensure that the Redis instance will be assigned a private IP rather than a public IP address.

**Note** Not all Azure Cache for Redis services support private link (4/2021).

**Note** There are many other [basic Azure Networking considerations](https://docs.microsoft.com/en-us/azure/Redis/concepts-data-access-and-security-vnet) that must be taken into account that are not the focus of this guide.

## Security baseline

Review a set of potential [security baseline](https://docs.microsoft.com/en-us/azure/Redis/security-baseline) tasks that can be implemented across all Azure resources. Not all of the items described on the reference link will apply to the specific data workloads or Azure resources.

## Security Checklist

* Use Azure AD authentication where possible.
* Enable Advanced Thread Protection.
* Enable all auditing features.
* Consider a Bring-Your-Own-Key (BYOK) strategy.
* Implement firewall rules.
* Utilize private endpoints for workloads that do not travel over the Internet.

# Summary

This document has covered several topics related to migrating an application from on-premises Redis to Azure Cache for Redis. We covered how to begin and assess the project all the way to application cut over.

The migration team will need to review the topics carefully as the choices made can have project timeline effects. The total cost of ownership is very attractive given the many enterprise ready features provided.

The migration project approach is very important. The team will need to assess the application and instance complexity to determine the amount of conversion time. Conversion tools will help make the transition easier, but there will always be an element of manual review and updates required. Scripting out pre-migration tasks and post migration testing is important.

Application architecture and design can provide strong indicators as to the level of effort required. For example, being able to add a layer of abstraction for moving cache data at runtime.

In the end, several tools exist in the marketplace ranging from free to commercial. This document covered the steps required if the team plans a instance migration using one of the more popular open source tool options. Whichever path that is chosen, Microsoft and the Redis community have the tools and expertise to make the instance migration successful.

## Questions and Feedback

For any questions or suggestions about working with Azure Cache for Redis, send an email to the Azure Cache for Redis Team (TODO@service.microsoft.com). Please note that this address is for general questions rather than support tickets.

In addition, consider these points of contact as appropriate:

* To contact Azure Support or fix an issue with your account, [file a ticket from the Azure portal](https://portal.azure.com/#blade/Microsoft_Azure_Support/HelpAndSupportBlade/overview).

## Find a partner to assist in migrating

This guide can be overwhelming, but don’t fret! There are many experts in the community with a proven migration track record. [Search for a Microsoft Partner](https://www.microsoft.com/solution-providers/home) or [Microsoft MVP](https://mvp.microsoft.com/MvpSearch) to help with finding the most appropriate migration strategy. You are not alone!

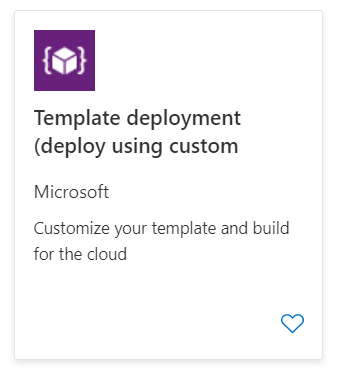
Browse the technical forums and social groups for more detailed real-world information:

* [Microsoft Community Forum](https://social.msdn.microsoft.com/forums/azure/en-US/home?forum=azurecache)
* [Microsoft Q&A Forum](https://docs.microsoft.com/en-us/answers/search.html?c=&includeChildren=&f=&type=question+OR+idea+OR+kbentry+OR+answer+OR+topic+OR+user&redirect=search%2Fsearch&sort=relevance&q=%5Bazure-cache-redis%5D)
* [StackOverflow for Azure Redis](https://stackoverflow.com/questions/tagged/azure-redis-cache)
* [Azure Facebook Group](https://www.facebook.com/groups/MsftAzure)
* [LinkedIn Azure Group](https://www.linkedin.com/groups/2733961/)
* [LinkedIn Azure Developers Group](https://www.linkedin.com/groups/1731317/)

# Appendix A: Environment Setup

The following steps will configure an environment to perform the guide’s migration steps.

## Deploy the ARM template

* Open the Azure Portal
* Create a new resource group
* Select **+Add**, type **template**, select the **Template Deployment…**
* 
* Select **Create**
* Select **Build your own template in the editor**
* Choose between the [secure](file:///D:\data\projects\redis-cache\pandoc\artifacts\template-secure.json) or the [non-secure](file:///D:\data\projects\redis-cache\pandoc\artifacts\template.json) ARM template. The difference between the two options is the secured option’s resources are hidden behind an App Gateway with private endpoints, whereas the other, resources are directly exposed to the internet.

TODO

**Note** The secure template runs at ~$2550 per month. The non-secure template runs at ~$1550 per month.

* Copy the json into the window
* Select **Save**
* 
* Fill in the parameters
  + Be sure to record your prefix and password, they are needed later
* Select **Review + create**
* Select the **I agree…** checkbox
* Select **Create**, after about 20 minutes the landing zone will be deployed

## Setup Redis Binding

* Login to the redis image
* Open the redis.conf file:

sudo nano /etc/redis/redis.conf

* Update the redis.conf, add the following bind statement below the others:

bind 0.0.0.0

## Setup Redis Cluster

* Run the following commands

sudo service redis-server restart

## Open the Azure VM Ports

TODO

* Browse to the Azure Portal.
* Select the **PREFIX-vm-redis01** virtual machine resource.
* Under **Settings**, select **Networking**
* In the **Inbound port rules** area, select **Add inbound port rule**
* For the **Destination port ranges**, type **5432**
* For the name, type **Port\_5432**
* Select **Add**

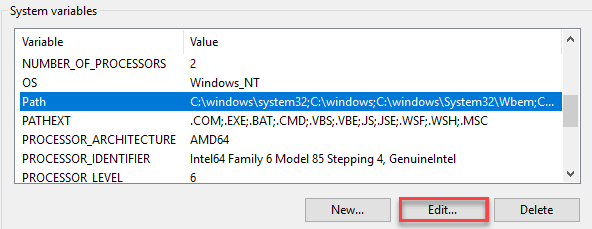
## Allow Azure Redis Access

* Browse to the Azure Portal.
* Select the **PREFIX-pg-single-01** instance.
* Under **Settings**, select **Connection security**
* Toggle the **Allow access to Azure services** to **On**
* Select **Save**
* Browse to your resource group
* Select the **PREFIX-pg-flex-01** instance.
* Under **Settings**, select **Networking**
* Toggle the **Allow public access from any Azure service within Azure to this server** to **On**
* Select **Save**

## Connect to the Azure VM

* Login to the deployed instance VM
  + Browse to the Azure Portal.
  + Select the **PREFIX-vm-dev** virtual machine resource.
  + Select **Connect->RDP**
  + Select **Open** in the RDP dialog
  + Login using s2admin and Seattle123Seattle123

## Install Redis 6.x

* In the Virtual Machine, download the following [Redis versions](https://www.enterprisedb.com/downloads/postgres-Redis-downloads)
  + [Redis 10.16](https://get.enterprisedb.com/Redis/Redis-10.16-1-windows-x64.exe)
  + Redis 11.0
* Install Redis 10.16
  + Start the Redis 10.16 installer you just downloaded
  + Select **Run**
  + On the Welcome dialog, select **Next**
  + On the installation directory dialog, select **Next**
  + On the select components dialog, select **Next**
  + On the data directory dialog, select **Next**
  + For the password, type Seattle123, then select **Next**
  + For the port, select **Next**
* *Note* the default port is 5432, if you have changed this port, you will need to ensure that you open the necessary paths in the firewall and gateways from Azure to your on-premisesenvironment.
  + Select your locale, select **Next**
  + On the summary dialog, select **Next**
  + On the ready dialog, select **Next**, Redis will start the installation process
  + Select **Finish**
* Add the \*\*C:Files\10\* path to the PATH environment variable
  + Switch to the Environment Variables window
  + Under **System variables**, choose **Path**. Then, select **Edit…**
  + 
  + In the **Edit environment variable** dialog, select **New** and then **Browse…** Browse to C:\Program Files\Redis\10\bin.
  + Select **OK**.

## Download artifacts

Perform the following on the **PREFIX-vm-pgdb01** virtual machine resource.

* Download and Install [Git](https://git-scm.com/download/win)
  + Download and run the 64-bit installer
  + Click **Next** through all prompts
* Open a Windows PowerShell window (just by entering “PowerShell” into the Start menu) and run the following commands

mkdir c:\Redisguide  
cd c:\Redisguide  
git config --global user.name "FIRST\_NAME LAST\_NAME"  
git config --global user.email "MY\_NAME@example.com"  
git clone https://github.com/solliancenet/onprem-postgre-to-azurepostgre-migration-guide

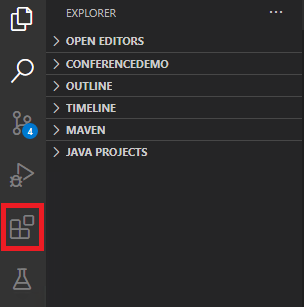
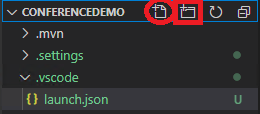
## Deploy the instance

Perform the following on the **PREFIX-vm-pgdb01** virtual machine resource.

* TODO

## Configure the Web Application API

Perform the following on the **PREFIX-vm-pgdb01** virtual machine resource.

* Open Visual Studio Code, if prompted, select **Yes, I trust the authors**
* Open the **C:-postgre-to-azurepostgre-migration-guide\* folder (Ctrl+K and Ctrl+O, or** File->Open Folder…\*\*)
* Select the **Extensions** tab
* 
* Search for and install the following extensions
  + Redis (by Microsoft)
  + Java Extension Pack
  + Spring Initializer Java Support
* When prompted, select **Yes** to trust the **Maven Wrapper**
* Update the .vscode\launch.json file
  + If a launch.json does not exist, create a .vscode folder, and then create a new file called launch.json. The rectangle highlights the tool used to create a new folder, while the oval indicates the tool to create a new file.
  + 
  + Copy the following into it:
  + {  
     // Use IntelliSense to learn about possible attributes.  
     // Hover to view descriptions of existing attributes.  
     // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387  
     "version": "0.2.0",  
     "configurations": [  
     {  
     "type": "java",  
     "name": "Debug (Launch)",  
     "request": "launch",  
     "mainClass": "com.yourcompany.conferencedemo.ConferencedemoApplication",  
     "env" :{  
     "DB\_CONNECTION\_URL" : "jdbc:Redis://localhost:5432/reg\_app?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC&noAccessToProcedureBodies=true",  
     "DB\_USER\_NAME" : "conferenceuser",  
     "DB\_PASSWORD" : "Seattle123",  
     "ALLOWED\_ORIGINS" : "\*",  
     }  
     }  
     ]  
    }
  + Update the **{DB\_CONNECTION\_URL}** environment variable to the Redis Connections string jdbc:Redis://localhost:5432/reg\_app?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC&noAccessToProcedureBodies=true
  + Update the **{DB\_USER\_NAME}** environment variable to the Redis Connections string conferenceuser
  + Update the **{DB\_PASSWORD}** environment variable to the Redis Connections string Seattle123
  + Update the **{ALLOWED\_ORIGINS}** environment variable to \*
* Select the **Debug** tab (directly above the **Extensions** tab from earlier), then select the debug option to start a debug session
* 
* If prompted, select **Yes** to switch to standard mode

## Test the Web Application

* Open a browser window, browse to **http://localhost:8888**
* Ensure the application started on port 8888 and displays results

## Configure the Web Application Client

* Open a new Visual Studio Code window to **C:-Redis-to-azureRedis-migration-guide-client**
* Open a terminal window (**Terminal**->**New Terminal**)
* Run the following commands to install all the needed packages, if prompted, select **N**

$env:Path = [System.Environment]::GetEnvironmentVariable("Path","Machine")  
  
npm install  
npm install -g @angular/cli

Note: If PowerShell indicates that npm is not a recognized command, try restarting VS Code.

* Close the terminal window and open a new one
* Run the following commands to run the client application

ng serve -o

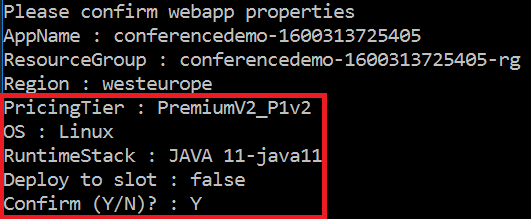
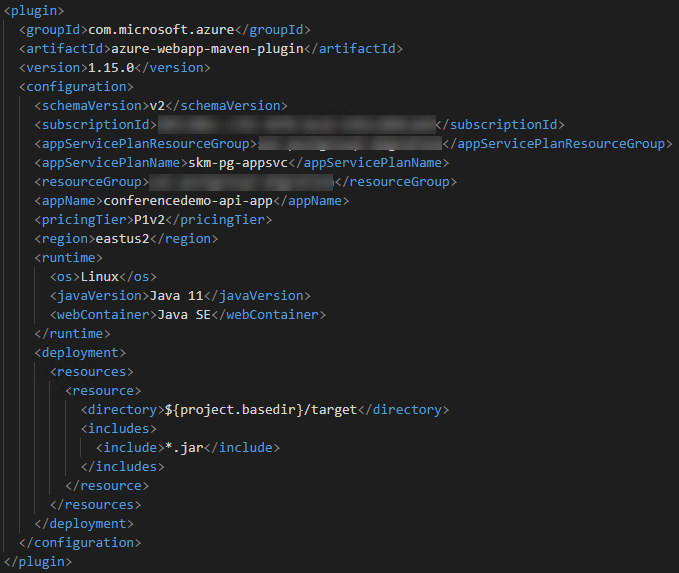
* A browser will open to the node site **http://localhost:{port}**
* Browse the conference site, ensure sessions and speaker pages load

**Note** If you don’t see any results, verify that the API is still running.

## Deploy the Java Server Application to Azure

* Open a command prompt window
* Run the following command to create the Maven configuration to deploy the app.
* Be sure to replace the maven version (ex 3.8.1)

cd C:\Redisguide\onprem-postgre-to-azurepostgre-migration-guide\artifacts\testapp\conferencedemo  
mvn com.microsoft.azure:azure-webapp-maven-plugin:1.15.0:config

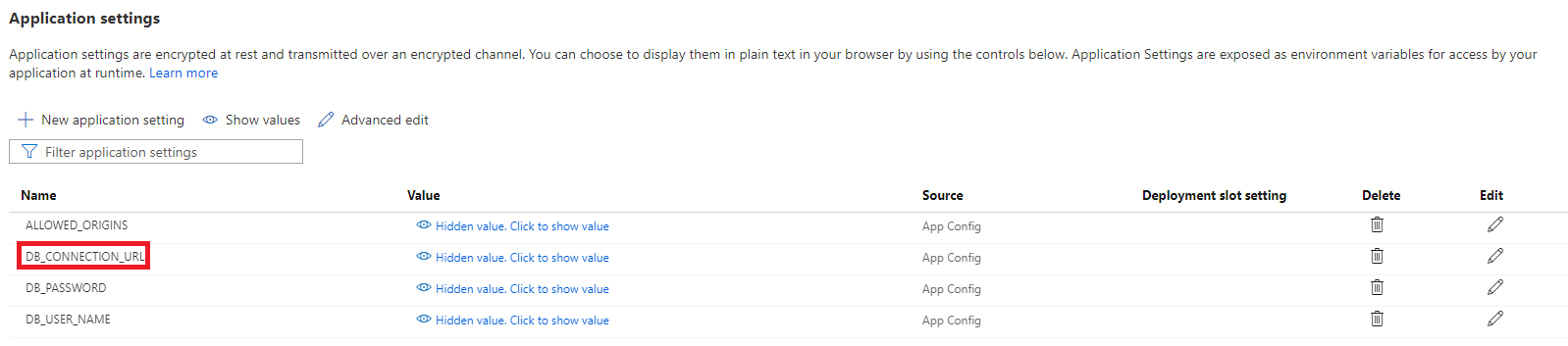
* Multiple packages will be installed from the Maven repository. Wait for the process to complete.
* When prompted, for the Define value for OS(Default:Linux), select the option that corresponds to linux or press **ENTER**
* Select Java 11
* Type **Y** to confirm the settings, then press **ENTER**
* 
* Switch to Visual Studio Code and the **ConferenceDemo** project
* Open the pom.xml file, notice the **com.microsoft.azure** groupId is now added
* Modify the resource group, appName and region to match the ones deployed in the sample ARM template
* **Note**: You may also need to specify the appServicePlanName and appServicePlanResourceGroup fields in the file, given that the ARM template already deploys an App Service plan. Here is an example of how the pom.xml file looks. We recommend using conferencedemo-api-app-[SUFFIX] as the appName.
* 
* If you have more than one subscription, set the specific subscriptionId in the [maven configuration](https://github.com/microsoft/azure-maven-plugins/wiki/Authentication#subscription)
  + Add the subscriptionId xml element and set to the target subscription
* If the secure landing zone has been deployed, set the hosts file
  + Browse to your resource group, select the **PREFIXapi01** app service
  + Select **Networking**
  + Select **Configure your Private Endpoint connections**
  + Select the **PREFIXapi-pe** private endpoint
  + Record the private IP Address
  + Repeat for the **PREFIXapp01** app service
  + Open a Windows PowerShell ISE window
  + Copy in the code from below, be sure to replace tokens, and save to **C:-Redis-to-azureRedis-migration-guide\* as** ConfiguringHostsFile.ps1\*\*
  + $prefix = "{PREFIX}";  
    $apiip = "{APIIP}";  
    $app\_name = "$($prefix)api01";  
      
    $hostname = "$app\_name.azurewebsites.net"  
    $line = "$apiip`t$hostname"  
    add-content "c:\windows\system32\drivers\etc\hosts" $line  
      
    $hostname = "$app\_name.scm.azurewebsites.net"  
    $line = "$apiip`t$hostname"  
    add-content "c:\windows\system32\drivers\etc\hosts" $line  
      
    $appip = "{APPIP}"  
    $app\_name = "$($prefix)app01";  
    $hostname = "$app\_name.azurewebsites.net"  
    $line = "$appip`t$hostname"  
    add-content "c:\windows\system32\drivers\etc\hosts" $line  
      
    $hostname = "$app\_name.scm.azurewebsites.net"  
    $line = "$appip`t$hostname"  
    add-content "c:\windows\system32\drivers\etc\hosts" $line
    - Run the file
    - 
* In the command prompt window from earlier, run the following to deploy the application. Be sure to replace the maven version (ex 3.8.1)

mvn package azure-webapp:deploy

* When prompted, login to the Azure Portal
* Update the App Service configuration variables by running the following, be sure to replace the tokens:

$prefix = "{PREFIX}";  
$app\_name = "$($prefix)api01";  
$rgName = "{RESOURCE-GROUP-NAME}";  
az login  
az account set --subscription "{SUBSCRIPTION-ID}"  
az webapp config appsettings set -g $rgName -n $app\_name --settings DB\_CONNECTION\_URL={DB\_CONNECTION\_URL}  
az webapp config appsettings set -g $rgName -n $app\_name --settings DB\_USER\_NAME={DB\_USER\_NAME}  
az webapp config appsettings set -g $rgName -n $app\_name --settings DB\_PASSWORD={DB\_PASSWORD}  
az webapp config appsettings set -g $rgName -n $app\_name --settings ALLOWED\_ORIGINS=\*

**Note** You will need to escape the ampersands in the connection string. You may consider inputting the value through Azure Portal as well. Navigate to the API App Service, and select **Configuration** under **Settings**. Then, under **Application settings**, manually enter the values.



* Restart the Java API App Service by running the following

az webapp restart -g $rgName -n $app\_name

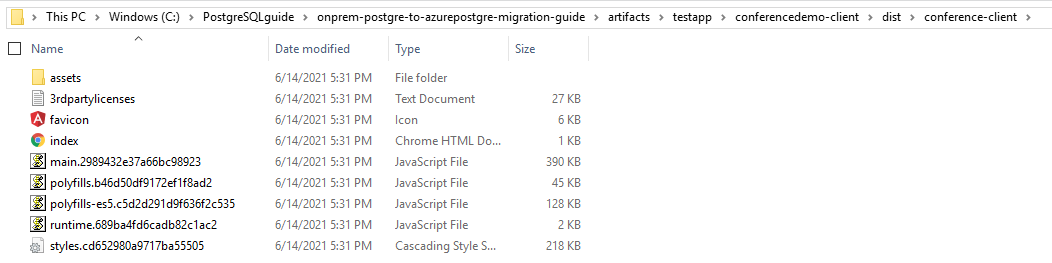
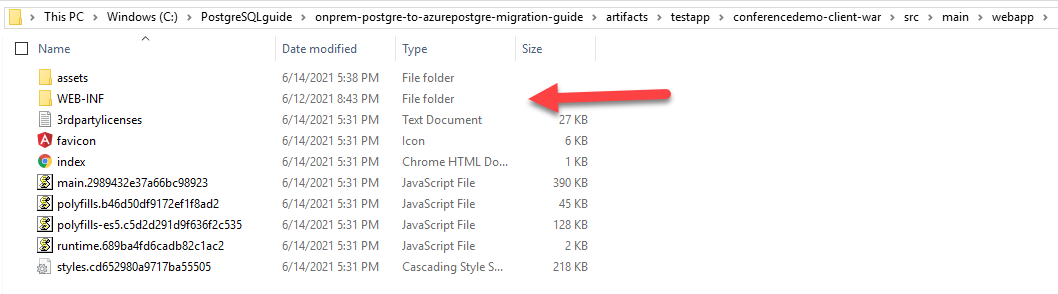
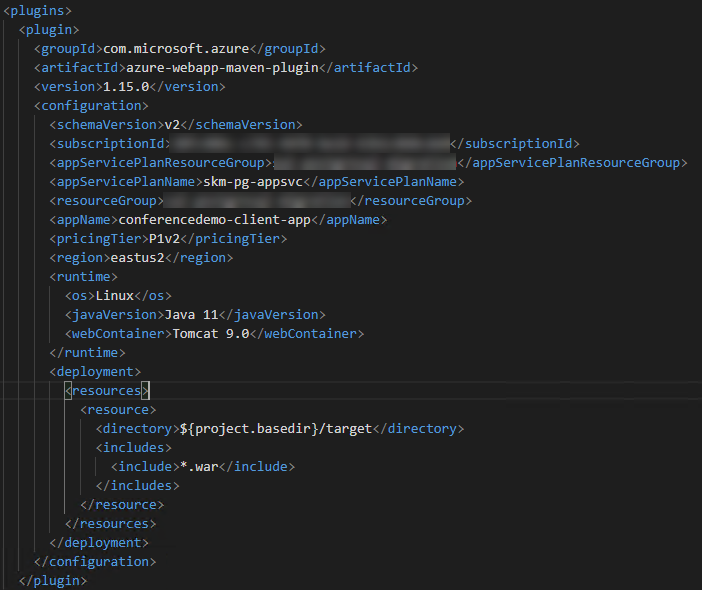
## Deploy the Angular Web Application to Azure

* Switch to the Visual Studio Code window for the Angular app (Conferencedemo-client)
* Navigate to **src.prod.ts**
* Set **webApiUrl** to **[JAVA APP SERVICE URL]/api/v1**

**Note** the App service url will come from the App Gateway service blade if using the secure deployment, or the App Service blade if not using the secure deployment.

* Run the following command to package the client app

ng build --configuration production

* Navigate to C:\Redisguide\onprem-Redis-to-azureRedis-migration-guide\artifacts\testapp\conferencedemo-client\dist\conference-client and copy the contents of that directory
* 
* This image demonstrates the artifacts of the production-ready, built Angular app.
* Open a new File Explorer window, and navigate to C:\Redisguide\onprem-postgre-to-azurepostgre-migration-guide\artifacts\testapp\conferencedemo-client-war\src\main\webapp. In this case, the built Angular app is packaged as a WAR archive, where it will be served by a Tomcat server in Azure App Service. Paste the contents copied in the previous step to this folder. Be mindful of the existing WEB-INF directory.
* 
* The built Angular app is added to the content root path of the WAR archive.
* Use your preferred text editor to open C:\Redisguide\onprem-postgre-to-azurepostgre-migration-guide\artifacts\testapp\conferencedemo-client-war\pom.xml. Take note of the azure-webapp-maven-plugin that has already been added for you.
* 
* azure-webapp-maven-plugin configured in the WAR POM file.
* Ensure that the following fields have been properly completed. Then, in command prompt (or the VS Code terminal), type mvn package azure-webapp:deploy.
  + **subscriptionId**: Your Azure subscription ID.
  + **appServicePlanResourceGroup**: The resource group used to complete this lab.
  + **appServicePlanName**: The App Service plan deployed with the ARM template (i.e. **PLACEHOLDER-pg-appsvc**)
  + **resourceGroup**: The resource group used to complete this lab.
  + **appName**: Provide a unique value, such as **conferencedemo-client-app-SUFFIX**.
* Once the deployment succeeds, navigate to the sample app in your browser. Confirm that everything works as anticipated.
* Congratulations. You have migrated the sample app to Azure. Now, focus on migrating a multi-tenant app to Azure to explore the power of horizontally-scalable Redis (Citus).

## Configure Network Security (Secure path)

* When attempting to connect to the instance from the app service, an access denied message should be displayed. Add the app virtual network to the firewall of the Azure Cache for Redis
  + Browse to the Azure Portal
  + Select the target resource group
  + Select the {PREFIX}Redis resource
  + Select **Connection security**
  + Select the Allow access to all Azure Services toggle to Yes
  + Select **Save**

# Appendix B: ARM Templates

## Secure

This template will deploy all resources with private endpoints. This effectively removes any access to the PaaS services from the internet.

[ARM Template](file:///D:\data\projects\redis-cache\pandoc\artifacts\template-secure.json)

## Non-Secure

This template will deploy resources using standard deployment where all resources are available from the internet.

[ARM Template](file:///D:\data\projects\redis-cache\pandoc\artifacts\template.json)

# SSL Connectivity with STunnel

In order to make calls to an SSL enabled version of Redis, you will need the latest redis-cli (6.x or higher) or you will need to enable a proxy to the SSL endpoints.

## Install STunnel

sudo apt-get install stunnel4 -y

* Setup the SSL port for Azure Redis

sudo nano /etc/default/stunnel4

* Set the Enabled value to 1
* Save the file
* Setup the pids directory and create the redis connection:

cd  
  
mkdir pids  
  
sudo chown -R nobody:nogroup pids/  
  
sudo nano /etc/stunnel/redis.conf

* Add the following:

[redis-cli]  
client = yes  
accept = 127.0.0.1:6380  
connect = yourcachename.redis.cache.windows.net:6380

* Restart stunnel

/etc/init.d/stunnel4 restart

* Ensure that stunnel is running:

sudo apt install net-tools  
  
sudo netstat -tlpn | grep 6380

# Configure Redis SSL

SSL/TLS is supported by Redis starting with version 6 as an optional feature that needs to be enabled at compile time.

## Download Redis 6.0

* Run the following to download Redis 6.0

wget https://download.redis.io/releases/redis-6.2.5.tar.gz  
tar xzf redis-6.2.5.tar.gz  
cd redis-6.2.5

## Build with TLS support

* Run the following to build Redis with TLS support:

sudo apt install make pkg-config libssl-dev  
  
make distclean  
  
make BUILD\_TLS=yes MALLOC=libc

## Create Certificates

* Create the self-signed certificates:

./utils/gen-test-certs.sh  
  
sudo apt-get install -y tcl-tls  
  
./runtest --tls

## Start the server

./src/redis-server --tls-port 6380 --port 6379 \  
 --tls-cert-file ./tests/tls/redis.crt \  
 --tls-key-file ./tests/tls/redis.key \  
 --tls-ca-cert-file ./tests/tls/ca.crt &

The Redis instance should be running and now ready to accept SSL connections.

## Test connectivity

./src/redis-cli --tls \  
 --cert ./tests/tls/redis.crt \  
 --key ./tests/tls/redis.key \  
 --cacert ./tests/tls/ca.crt

## Resources

* https://redis.io/topics/encryption