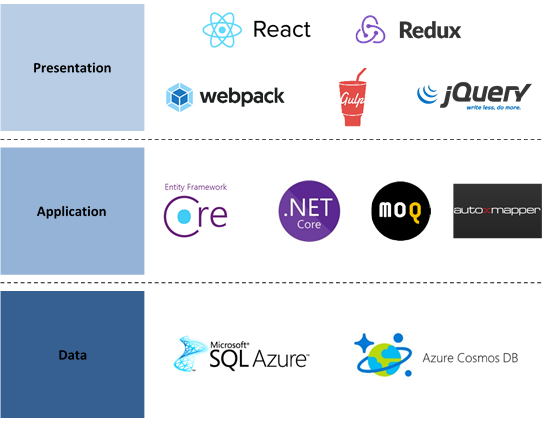
**Architecture**

**Technology Stack**

**3.2.1 Standard**

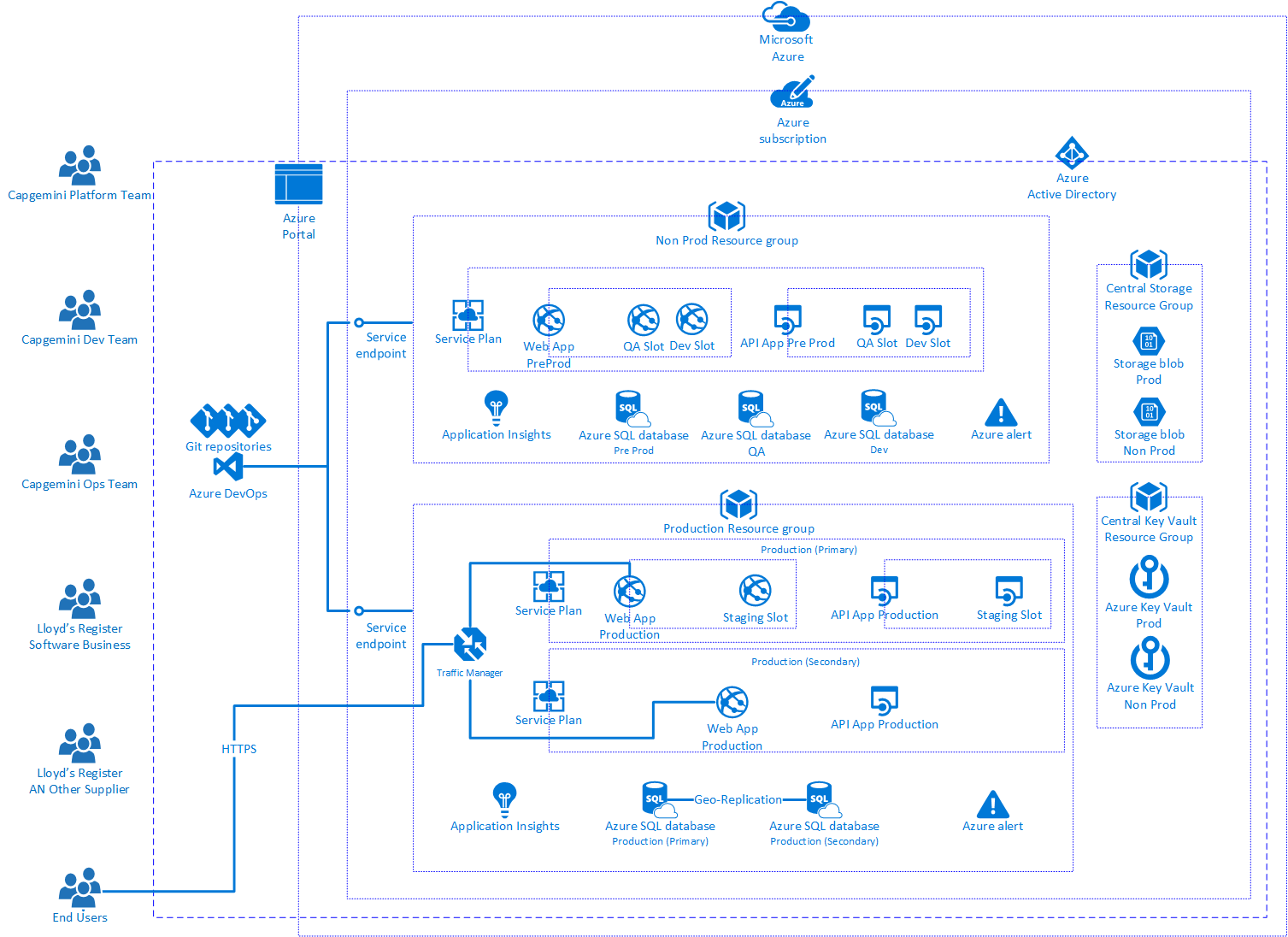


**3.2.2 Alternative Technology Stacks**

Azure App Service allows the creation of Web apps and Api apps using alternative technology stacks, supporting the following; [ASP.NET](http://asp.net/) , Java, Ruby, Node.js, PHP, or Python.

This reference architecture will provide a basic template for the deployment of an [ASP.NET](http://asp.net/) Core Web Application into the Azure Resources that are deployed. However, an empty template will also be provided to create an application using one of these alternatives.

## Physical Components



All components that are part of a resource group will be created by a collection of ARM Templates. The ARM templates are parameters driven to enable reuse across various projects.

Depending upon the requirements of a project Resource Groups will be created for Non-Production and Production for each Application. Some project may only request the creation of a Production Environment where they control non-Prod environment outside the platform.

A general Small PaaS application resource group will contain the following resources:

• Azure Traffic Manager.  
• Web App  
• API App  
• App Service Plan  
• SQL DB  
• SQL Server  
• App Insights

The platform also relies upon a series of other Azure Resources that sit in separate resource groups, such as Central Storage and Central Key Vault

All resources will be appropriately named and tagged for easy identification and billing management.

The following services are part of the deployment process:  
• Azure DevOps (Formerly Visual Studio Team Services or VSTS)  
• Service Endpoint/Connections  
• GitHub Repository

A resource group will be created per project, the resources within the Resource Group will be tagged with the environment name as part of the deployment of the resources. These tags can then be used for billing purposes to show cost at an environment level

All primary resources will be deployed in LR’s standard Region; North Europe with any secondary resource deployed by Default to West Europe unless otherwise requested by the application owner. This may be required to achieve better results where end users are based in other global regions i.e., US, Asia, etc.

## Application

The application components for this solution are split between a Web App and an API App. They both utilise a single Application Service Plan for defining the Pricing Tier and Auto Scaling Rules.

An App service plan will be created for each of the necessary application environments, i.e. Development, QA, Pre-Production and Production. The respective Web App and Api App will then be associated with the correct App Service Plan. The currently available App Service plans are as follows:  
The pricing tier of an App Service plan determines what App Service features you get and how much you pay for the plan.

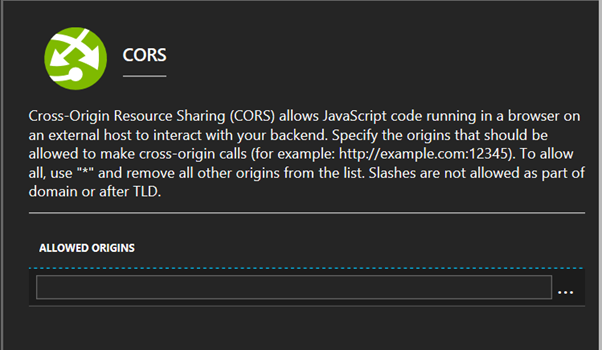
• **Shared compute**: Free and Shared, the two base tiers, runs an app on the same Azure VM as other App Service apps, including apps of other customers. These tiers allocate CPU quotas to each app that runs on the shared resources, and the resources cannot scale out.  
• **Dedicated compute**: The Basic, Standard, Premium, and PremiumV2 tiers run apps on dedicated Azure VMs. Only apps in the same App Service plan share the same compute resources. The higher the tier, the more VM instances are available to you for scale-out. Although only Standard and Premium allow automatic scale-out features.  
**• Isolated**: This tier runs dedicated Azure VMs on dedicated Azure Virtual Networks, which provides network isolation on top of compute isolation to your apps. It provides the maximum scale-out capabilities.

In the Shared tier, each app receives a quota of CPU minutes, so each app is charged hourly for the CPU quota.  
**In the dedicated** compute tiers (Basic, Standard, Premium, PremiumV2), The App Service plan defines the number of VM instances the apps are scaled to, so each VM instance in the App Service plan has an hourly charge. These VM instances are charged the same regardless how many apps are running on them. To avoid unexpected charges, see Clean Up an App Service plan.  
**In the Isolated tier**, the App Service Environment defines the number of isolated workers that run your apps, and each worker is charged hourly. In addition, there's an hourly base fee for the running the App Service Environment itself.

The Web App will host the data management application and will be comprised of an MVC Application.

The API App will provide services to external applications that require access to specific data from the system. Access to these services will be governed and restricted through the Api Management Service.

In order to allow the Web App to make JavaScript client-side AJAX calls to the API App APIs it will be necessary to enable Cross-Origin Resource Sharing by adding the WebApp URL to the CORS Allowed Origin of the API Application within the Azure Portal.



## Information

### 3.5.1 Relational Data Storage

The Relational Data for this solution will utilise Azure SQL Database. Azure SQL Database is a relational database-as-a service using the Microsoft SQL Server Engine. This offering has a number of service tiers, information on the tiers and features included can be found here: <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-service-tiers> .

A connection String for the database will be automatically created within the Application Settings of the Web App on deployment.

In most cases it will be necessary to evaluate the cost of DBaaS offering of Azure SQL against a traditional database implementation. The pricing for Azure SQL can be found here: <https://azure.microsoft.com/en-gb/pricing/details/sql-database/> . The various resource limitations can be found here: <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-resource-limits> .

### 3.5.2 Non-Relational Data Storage

3.5.2.1 Storage Account

A central storage account will be used to hold all logs and backups created by the application and database. The storage can also be used for holding any other files required by the application e.g. Images, Media files, etc. Instructions on how to add an retrieve files within the application can be found here: <https://docs.microsoft.com/en-us/azure/storage/blobs/storage-dotnet-how-to-use-blobs> .

Due to limitations on the number of Storage accounts that can be created per subscription, it will be necessary to utilise a central storage account within the LR Subscription. A number of containers related to the application/environment/resource will be required. This will allow all logs, backups and application related files to be easily identified. Creation of the Containers in one of the existing Storage Accounts will need to be performed via the Azure Portal or Azure Storage Explorer Application. There is no limit on the number of containers that can be created within a Storage Account.

A standard set of containers should be created for each of the following per application and environment:  
• SQL Database Audit logs  
• Application Logs  
• Application General Files

**Security**

**3.6.1 Identity and Access Management**

**3.6.1.1 Azure Active Directory**

Azure Active Directory will be used as the primary authentication and authorisation method for any applications that are created using this reference architecture. All Applications that are deployed will need to be registered within the AAD Instance as a Registered Application. Once registered it will be necessary to create the Client ID and Client Secret for your application. The Authentication/Authorization can be setup for the application once it has been deployed via the resource manager template. The registration an be completed as part of the process or before.

Select App Service Authentication: On, then select Log in with Azure Active directory from the drop-down menu.

If new user groups are required to govern access to certain areas of the application, then the groups will need to be created by the AAD admin and calls to the Azure Graph API to retrieve the group information for the user will need to be included within the application coding.  
Link to Graph API Documentation: <https://developer.microsoft.com/en-us/graph/docs/concepts/aspnetmvc>

**3.6.1.2 Alternative - Azure B2C**

Although, Internal access had only been considered as part of this reference architecture, an alternative approach to Authorisation and Authentication can be used that will allow the self-service of external customer to LR.

**Network Security**

SQL databases are protected by a firewall in Azure. By default, all connections to the server and the databases inside the server are rejected except for connections from other Azure services.

**3.6.2.1 SQL Server firewall**

Select the SQL Server from the list of resources within the resource group. The overview page for your SQL server opens, showing you the fully qualified server name (such as [lrdb.database.windows.net](http://lrdb.database.windows.net/) ) and provides options for further configuration. Click Set server firewall on the toolbar as shown in the previous image. The Firewall settings page for the SQL Database server opens.

Click Add client IP on the toolbar to add the public IP address of the computer connected to the portal with or enter the firewall rule manually and then click Save. Click OK and then click the X to close the Firewall settings page. You can now connect to any database in the server with the specified IP address or IP address range.

If the data classification requires network access to be restricted this method should be used to allow access to Azure resources and the default property of allowing access to all Azure Services should be disabled.

**3.6.2.2 SQL Database firewall**

Similar to the above SQL Server firewall the individual SQL database can be protected by a firewall. Where multiple databases are hosted on the same server this can be useful to add protection at a more granular level. However, for the purposes of this solution the SQL Databases are hosted on separate servers to allow for greater flexibility in the pricing tiers to be used for each of the environments.

**3.6.2.3 App Service Network Security**

This reference architecture does not recommend VNet Integration due to the need to include the app within an isolated App Service Environment deployment. These features are only required when implementing application that require integration with existing On-Premise resources. If these options are required more details can be found here: <https://docs.microsoft.com/en-us/azure/app-service/web-sites-integrate-with-vnet> and <https://docs.microsoft.com/en-us/azure/app-service/environment/intro> . Only App Service Environment supports the use of ExpressRoute.

Azure App Service restricts traffic to Web apps only on ports 80 and 443. Therefore the need for an external firewall is negated. As a standard all applications will enforce the use of HTTPS traffic to the site, further restricting access to only port 443. If the application require further restrictions to this then Client IP Restrictions can be included on the individual app service resource which will restrict access to all IP’s except those defined.  
<https://docs.microsoft.com/en-us/azure/app-service/app-service-ip-restrictions>

The use of TLS 1.2 is mandated.

LR already have a working Express Route connection into Azure from Fenchurch Street (i.e. routing and firewall already configured). The Express Route connection exists within the ‘lr\_internal\_net\_vnet’ virtual network which is contained within the ‘lr\_internal\_network’ resource group.

## Back-ups

### 3.7.1 Application Back-up

Once the app service and storage account have been deployed the backup policy can be created for each application. A policy can be created specifying the schedule, retention period and storage account that should be used.  
Be aware that the backed-up files include app settings in plain text and these may include secrets, such as connection strings. If this is not desired, then these back-ups should not be enabled.

### 3.7.2 Database Back-up

Depending on the Service Tier chosen for the SQL Database automatic backups are created with different retention periods; i.e. Standard Service Tier allows restore for up to 35 Days. However, it’s is also possible to create long-term retention backups for the Databases on an SQL Server. This is not available in all Azure Regions and should not be used if possible.

## Monitoring and Alerting

### 3.8.1 Application Monitoring

See platform Wiki - - [Application Logging and Monitoring](https://lloydsregistergroup.visualstudio.com/Digital%20Platform/_wiki/wikis/Digital-Platform.wiki?wikiVersion=GBwikiMaster&pagePath=/Digital%20Platform%20Wiki/Application%20Logging%20and%20Monitoring)

It is also possible to enable Alert and Application Insights on you your application.

### 3.8.2 Database Monitoring

#### 3.8.2.1 Audit

See platform Wiki - - [Application Logging and Monitoring](https://lloydsregistergroup.visualstudio.com/Digital%20Platform/_wiki/wikis/Digital-Platform.wiki?wikiVersion=GBwikiMaster&pagePath=/Digital%20Platform%20Wiki/Application%20Logging%20and%20Monitoring)

#### 3.8.2.2 Threat Detection

It is also possible to enable threat detection for the Database although this does introduce additional cost ($15/month/server).  
Treat Detection can be set on three types of threat:  
• SQL Injection  
• SQL Injection Vulnerability  
• Anomalous Client Login  
Any instances of these threats can then be sent as an Alert email to a specified Email Address.

This feature should be enabled as a minimum on the Production SQL Server.

## Availability

At the time of writing, the service level agreement (SLA) for App Service is 99.95% and the SLA for SQL Database is 99.99% for Basic, Standard, and Premium tiers.

#### 3.9.1  Azure Traffic Manager

Azure Traffic Manager is a DNS-based traffic load balancer that enables you to distribute traffic optimally to services across global Azure regions, while providing high availability and responsiveness.

Traffic Manager uses DNS to direct client requests to the most appropriate service endpoint based on a traffic-routing method and the health of the endpoints. An endpoint is any Internet-facing service hosted inside or outside of Azure. Traffic Manager provides a range of [traffic-routing methods](https://docs.microsoft.com/en-us/azure/traffic-manager/traffic-manager-routing-methods) and [endpoint monitoring options](https://docs.microsoft.com/en-us/azure/traffic-manager/traffic-manager-monitoring)to suit different application needs and automatic failover models. Traffic Manager is resilient to failure, including the failure of an entire Azure region.

it enabled planned maintenance operations on your applications without downtime. Traffic Manager directs traffic to alternative endpoints while the maintenance is in progress.

## Deployments

The ARM Templates will be available within the Digital Platform GitHub hosted within Azure DevOps.

When providing parameters for the ARM templates the Naming Standard for each resource type should be adhered to and are defined in the Naming Standards section at 3.11 below.

Deployment of the Template to Azure can be achieved using two methods. The use of Visual Studio Team Services with PowerShell where necessary is the preferred method:

### 3.10.1 Azure DevOps

Within VSTS a Service Endpoint is required for deployment into Azure. This Service Endpoint is created within VSTS as an Azure Resource Manager Service Endpoint.

Once configured a build can then be created within VSTS for the deployment of the ARM Template using the Azure Deployment: Create or Update Resource Group action on.

## Naming Standards

Resource Type Naming Standard  
**App Service Plan lr-<ApplicationName>-serviceplan-<Environment>  
App Service – Web App\* lr-<ApplicationName>-<Environment>  
App Service – Api App\* lr-<ApplicationName>-api-<Environment>  
SQL Server lr-<ApplicationName>-sqlserver-<Environment>  
SQL Database lr-<ApplicationName>-db-<Environment>  
Storage Account lr<ApplicationName><Environment>storage  
API Management Service lr-<ApplicationName>-api-management**  
\*The name applied to this resource is also used for the [azurewebsite.net](http://azurewebsite.net/) URL generated for the Application.

**Cosmos DB lr-<ApplicationName>-doc-db-<Environment>**

**Mobile App lr-<ApplicationName>-mobileapp-<Environment>**

**Notification Hub lr-<ApplicationName>-notificationhub**

**Function lr-<ApplicationName>-<module>-<Environment>**

**Code repository Branch details:**

There are three branches generally created for project.

1. Development branch - Developer works on and checks-in code to release branch.
2. Master branch - Acts as a base
3. Release branch - Used for deployment. CI/CD pipeline uses this branch.

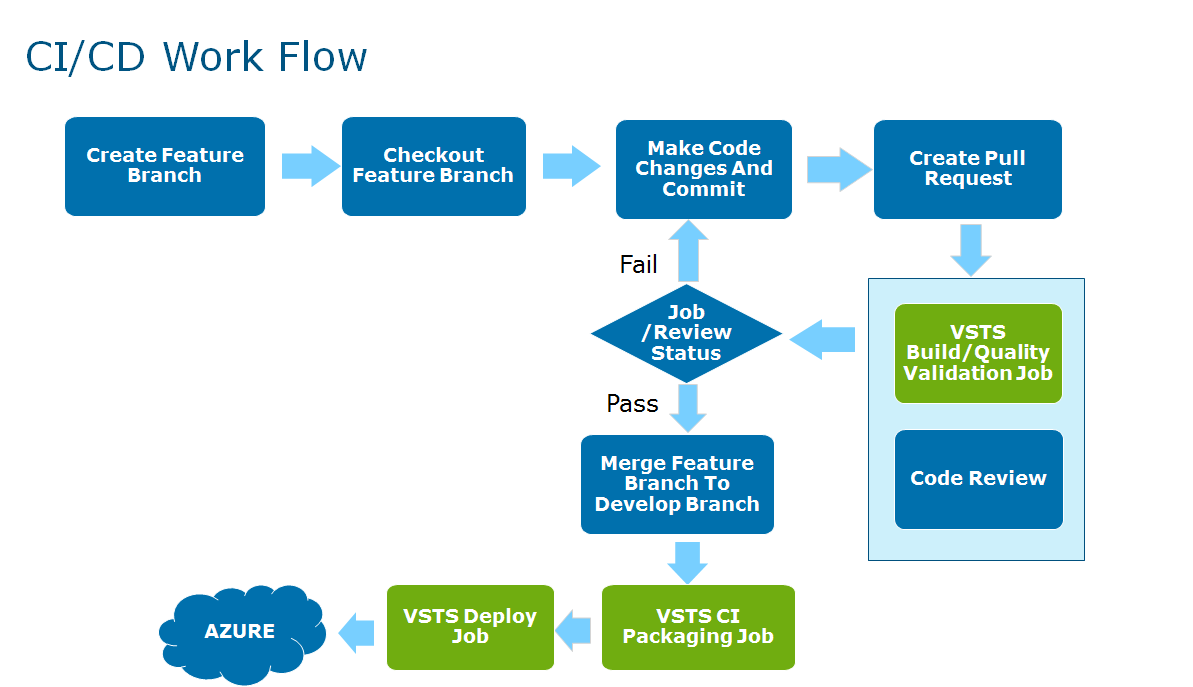
Currently LR project has below environments to deploy the code.  
**NonProd:** Before the code is deployed in Prod environment, the code will be tested in NonProd environment. This environment again classified into three environments. 1. Dev 2. QA 3. PreProd

**Prod:** This environment is where application is published to external world. This environment is again classified into two environments. 1. Staging 2. Prod.

**Pipelines Approvals Approach**

1. In Azure DevOps, for every pipeline digital platform owner is made as first approver.
2. For any intermediate stages digital platform owner is removed as the approver.

CICD workflow



**Logging and Monitoring**

**Introduction**

To monitor and / or debug applications logging and monitoring will be enabled by default.

**Log Location**

All logs are stored as blobs within Azure Storage (not filesystem).

|  |  |
| --- | --- |
| **Environment** | **Storage Account** |
| Production | lrappstorageprod |
| Pre-Production | lrappstoragenonprod |

Within these storage accounts are containers for the different log types:

|  |  |
| --- | --- |
| **Container** | **Log Type** |
| applogs*-<appname>-<environment>* | Application logging containing diagnostic traces from web apps and diagnostic information from the web server (aka IIS logs) |
| sqldbauditlogs | All SQL server audit logs |

e.g. applogs-allassets-staging

Log Level and Retention

Below are the details of the logging that will be implemented by default for normal operation.

The level of reporting can be 'turned up' temporarily to enable diagnosis of any issues. This must be formally requested of the Digital Platform team by the support team

**Application Logging**

* Error only
* 365 days retention

**Web Server Logging**

Web Server logs can be significant, hence the default is:

* 365 day retention
* Detailed error messages: Off
* Failed request tracing: Off

**SQL Server Logging**

* 365 days retention
* Audit Logging : On
* Threat Detection: Off

Threat detection carries an additional cost of $15 a month so it is turned off by default. However, where the data is particularly sensitive and there is a supporting requirement it can be turned-on manually.

**Access to Logs**

Access to the application logs will be granted to the relevant application support team only via their Active Directory Group.  
Access to the SQL Audit logs would not normally be provided as it allows access to all logs for all SQL server and database. However, it is suggested that certain logs could be copied, on demand, by the Digital Platform Team.

**Cosmos DB Backup and Restore**

**CosmosDB**

Azure Cosmos DB automatically takes backups of all data at regular intervals.

The automatic backups are taken without affecting the performance or availability of database operations. All the backups are stored separately in another storage service, and those backups are globally replicated for resiliency against regional disasters.

The automatic backups are intended for scenarios when Cosmos DB is accidentally deleted container and later require data recovery.  
  
**High availability with Cosmos DB**

Cosmos DB is designed to be globally distributed – it allows to scale throughput across multiple Azure regions along with policy driven failover and transparent multi-homing APIs.  
Azure Cosmos DB offers 99.99% availability SLAs for all single region accounts and all multi-region accounts with relaxed consistency, and 99.999% read availability on all multi-region database accounts.  
All the writes in Azure Cosmos DB are durably committed to local disks by a quorum of replicas within a local data center before acknowledging to the client.  
The high availability of Cosmos DB relies on local storage and does not depend on any external storage technologies. Additionally, if your database account is associated with more than one Azure region, the writes are replicated across other regions as well. To scale throughput and access data at low latencies, it enables as many read regions associated with your database account .In each read region, the (replicated) data is durably persisted across a replica set.

As illustrated in the following diagram, a single Cosmos DB container is horizontally partitioned. A "partition" is denoted by a circle in the following diagram, and each partition is made highly available via a replica set.  
This is the local distribution within a single Azure region (denoted by the X axis). Further, each partition (with its corresponding replica set) is then globally distributed across multiple regions associated with your database account (for example, in this illustration the three regions – East US, West US and Central India).  
The "partition set" is a globally distributed entity comprising of multiple copies of your data in each region (denoted by the Y axis). Priority can be assigned to the regions associated with database account and Cosmos DB will transparently fail over to the next region in case of disaster. Failovers can be manually stimulated to test the end-to-end availability of the application.

**Full, automatic and online backups**  
  
With Cosmos DB, not only the data, but the backups of the data are also made highly redundant and resilient to regional disasters. These automated backups are currently taken approximately every four hours and latest two backups are stored at all times. If the data is accidentally dropped or corrupted, contact [Azure support](https://azure.microsoft.com/support/options/) within eight hours.  
  
The backups are taken without affecting the performance or availability of the database operations. Cosmos DB takes the backup in the background without consuming the provisioned RUs or affecting the performance and without affecting the availability of the database.  
  
 **Backup Retention Period**

Azure Cosmos DB takes snapshots of the data every four hours at the partition level. At any given time, only the last two snapshots are retained. However, if the container/database is deleted, Azure Cosmos DB retains the existing snapshots for all of the deleted partitions within the given container/database for 30 days.

**Restoring a database from an online backup**

If database or container is accidentally deleted,  [file a support ticket](https://portal.azure.com/?#blade/Microsoft_Azure_Support/HelpAndSupportBlade) or [call Azure support](https://azure.microsoft.com/support/options/) to restore the data from the last automatic backup. Azure support is available for selected plans only such as Standard, Developer, support isn't available with Basic plan.

**Restore**

SQL Database provides these options for database recovery using [automated database backups](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-automated-backups) .

You can restore from a database backup to:

* A new database on the same logical server recovered to a specified point in time within the retention period.
* A database on the same logical server recovered to the deletion time for a deleted database.
* A new database on any logical server in any region recovered to the point of the most recent daily backups in geo-replicated blob storage (RA-GRS).

**Point-in-time restore.**

we can restore an existing database to an earlier point in time as a new database on the same logical server using the Azure portal, [PowerShell](https://docs.microsoft.com/powershell/module/azurerm.sql/restore-azurermsqldatabase), or the [REST API](https://msdn.microsoft.com/library/azure/mt163685.aspx).

we generally restore a database to an earlier point for recovery purposes. When doing so, you can treat the restored database as a replacement for the original database or use it to retrieve data from and then update the original database.

* ***Database replacement:*** If the restored database is intended as a replacement for the original database, you should verify the performance level and/or service tier are appropriate and scale the database if necessary. You can rename the original database and then give the restored database the original name using the [ALTER DATABASE](https://docs.microsoft.com/en-us/sql/t-sql/statements/alter-database-azure-sql-database) command in T-SQL.
* ***Data recovery:*** If the plan is to retrieve data from the restored database to recover from a user or application error, you need to write and execute the necessary data recovery scripts to extract data from the restored database to the original database. Although the restore operation may take a long time to complete, the restoring database is visible in the database list throughout the restore process. If you delete the database during the restore, the restore operation is canceled and you are not charged for the database that did not complete the restore.

**Deleted database restore.**

we can restore a deleted database to the deletion time for a deleted database on the same logical server using the Azure portal, [PowerShell](https://docs.microsoft.com/powershell/module/azurerm.sql/restore-azurermsqldatabase), or the [REST (createMode=Restore)](https://msdn.microsoft.com/library/azure/mt163685.aspx). You can restore a deleted database to an earlier point in time during the retention using [PowerShell](https://docs.microsoft.com/powershell/module/azurerm.sql/restore-azurermsqldatabase).

To recover a deleted database during its [DTU-based model retention period](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-service-tiers-dtu) or [vCore-based model retention period](https://docs.microsoft.com/en-us/azure/sql-database/sql-database-service-tiers-vcore) using the Azure portal, open the page for your server and in the Operations area, click **Deleted databases**.

**Geo-restore**

* A SQL database can be restored on any server in any Azure region from the most recent geo-replicated full and differential backups.
* Geo-restore uses a geo-redundant backup as its source and can be used to recover a database even if the database or datacenter is inaccessible due to an outage.
* Geo-restore is the default recovery option when the database is unavailable because of an incident in the region where the database is hosted.
* If a large-scale incident in a region results in unavailability of the database application, the database can be restored from the geo-replicated backups to a server in any other region.
* There is a delay between when a differential backup is taken and when it is geo-replicated to an Azure blob in a different region.
* This delay can be up to an hour, so, if a disaster occurs, there can be up to one hour data loss.

**The following illustration shows restore of the database from the last available backup in another region.**

* Point-in-time restore on a geo-secondary is not currently supported.
* Point-in-time restore can be done only on a primary database.

**Azure portal**

* To geo-restore a database during its DTU-based model retention period or v-Core based model retention using the Azure portal, open the SQL Databases page and then click Add.
* In the Select source text box, select Backup.
* Specify the backup from which to perform the recovery in the region and specify the server.

**Programmatically performing recovery using automated backups**

In addition to the Azure portal, database recovery can be performed programmatically using Azure PowerShell or the REST API. The following tables describe the set of commands available.

**PowerShell**

| Cmdlet | Description |
| --- | --- |
| [Get-AzureRmSqlDatabase](https://docs.microsoft.com/en-us/powershell/module/azurerm.sql/get-azurermsqldatabase) | Gets one or more databases. |
| [Get-AzureRMSqlDeletedDatabaseBackup](https://docs.microsoft.com/en-us/powershell/module/azurerm.sql/get-azurermsqldeleteddatabasebackup) | Gets a deleted database that you can restore. |
| [Get-AzureRmSqlDatabaseGeoBackup](https://docs.microsoft.com/en-us/powershell/module/azurerm.sql/get-azurermsqldatabasegeobackup) | Gets a geo-redundant backup of a database. |
| [Restore-AzureRmSqlDatabase](https://docs.microsoft.com/en-us/powershell/module/azurerm.sql/restore-azurermsqldatabase) | Restores a SQL database. |

Automatic backups protect your databases from user and application errors, accidental database deletion, and prolonged outages. This built-in capability is available for all service tiers and performance levels.

**Platform Insights**

**Application Monitoring.**

Digital platform is using two types of dashboard one is targeting platform and operation related data i.e. AppInsights dashboards and second one is mostly related to application development, CI, CD and testing related data i.e. VSTS  
dashboard per projects.

**AppInsights.**

Application Insights is an extensible Application Performance Management (APM) service for web developers on multiple platforms. Use it to monitor your live web application. It will automatically detect performance anomalies. It includes  
powerful analytics tools to help you diagnose issues and to understand what users do with your app. It's designed to help you continuously improve performance and usability. It integrates with your DevOps process, and has connection points to a variety of development tools.

**Dashboards.**

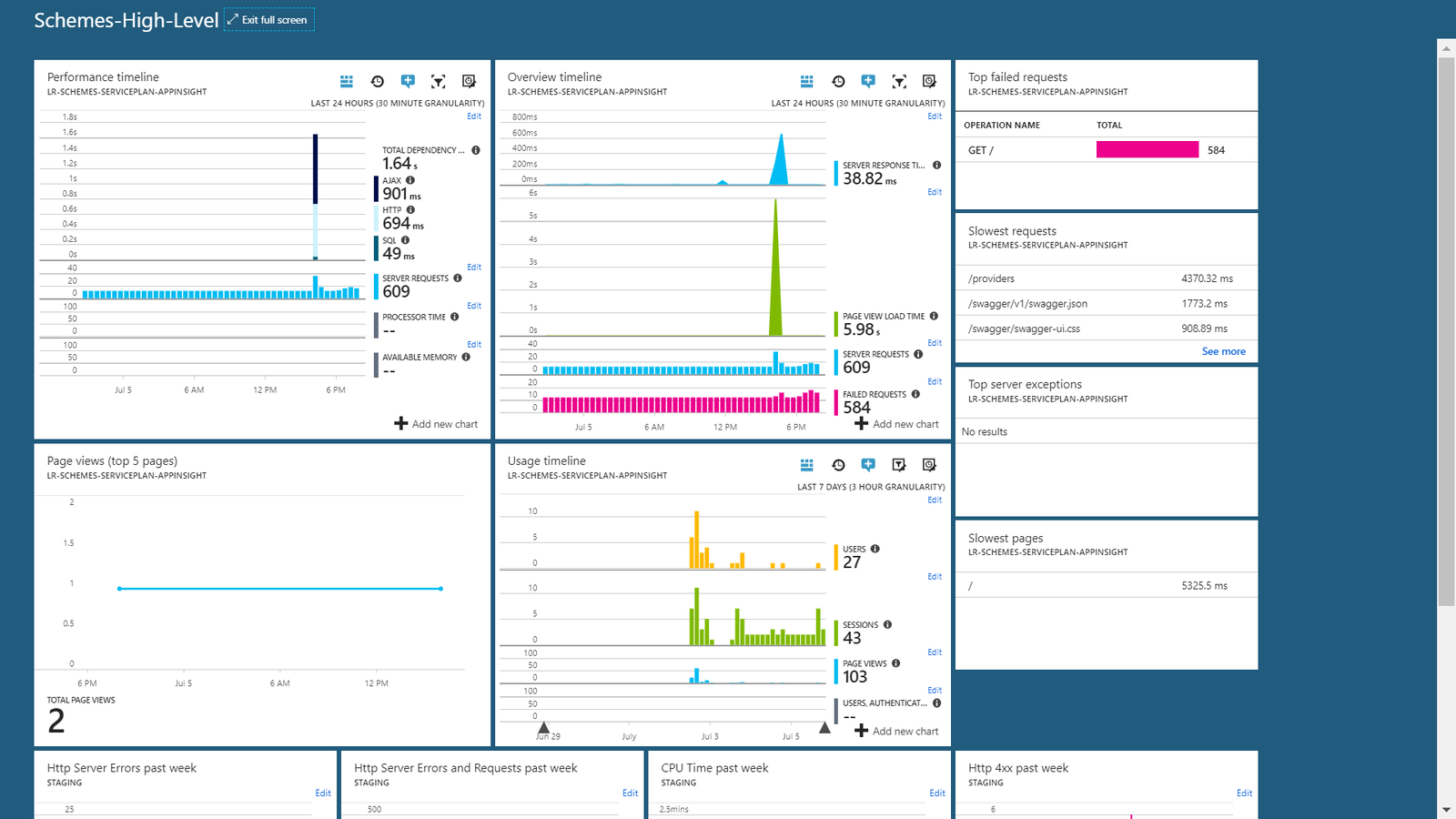
As per digital platform standard we have divided service dashboard into two parts:

**Views for external stakeholders**

* High-Level dashboards

Above type of dashboards target the high-level audience related to infrastructure and operations like Head of Operations and Head of Infrastructure with longer tenure approx. 1-week data should targeted for charting and monitoring with AppInsights data.

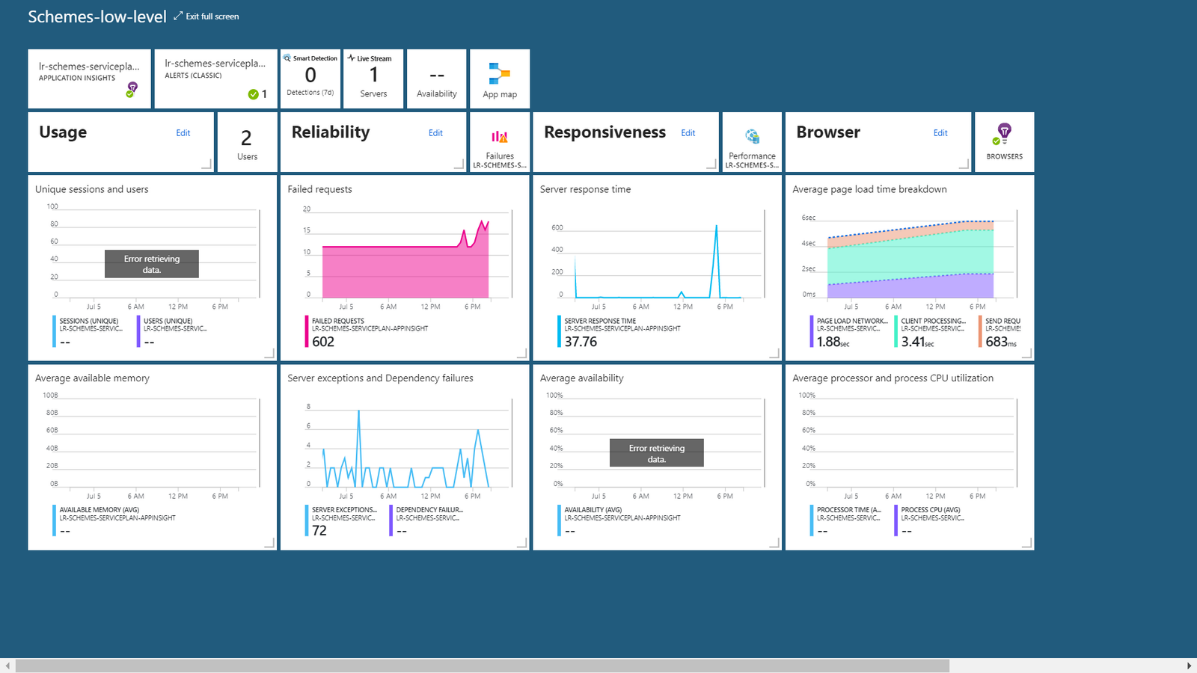
For Example:-



* Low-level dashboards

Above type of dashboards target the low-level audience related to infrastructure and operations like Product Owner and Head of architecture with shorter tenure approx. 24-hour data should targeted for charting and monitoring with extended views with AppInsights data.

for Example :-



**Example Views**

Below are some views that appear by default on the application blade.

**External Stakeholders - High-Level & Low Level dashboards:**

The below are performance monitor metrics of application.

**Requests**

The number of HTTP requests received in a specified period. Compare this with the results on other reports to see how your app behaves as the load varies.

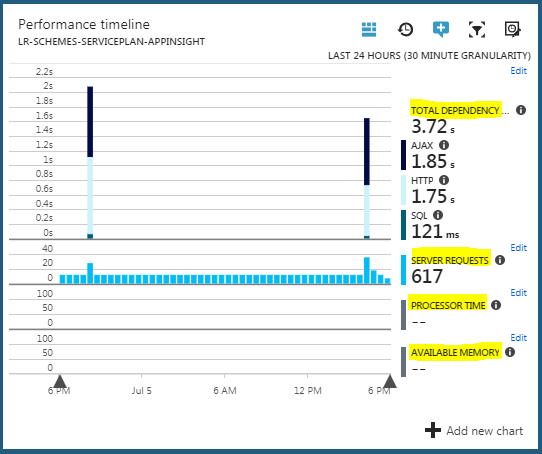
HTTP requests include all GET or POST requests for pages, data, and images.

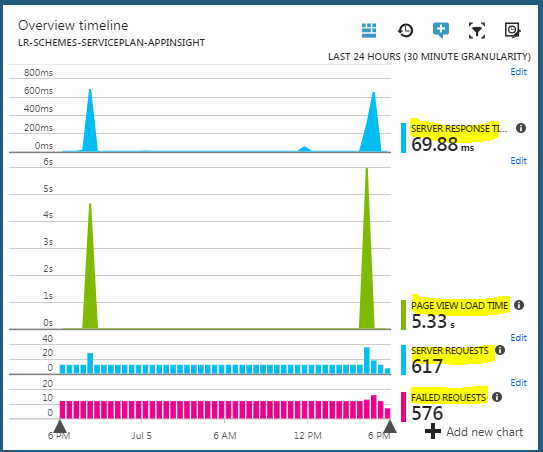
**Average response time**

Measures the time between a web request entering your application and the response being returned.

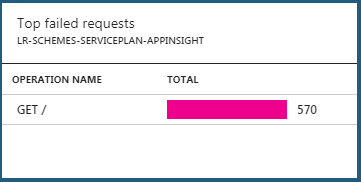
The points show a moving average. If there are many requests, there might be some that deviate from the average without an obvious peak or dip in the graph.

Look for unusual peaks. In general, expect response time to rise with a rise in requests. If the rise is disproportionate, your app might be hitting a resource limit such as CPU or the capacity of a service it uses.

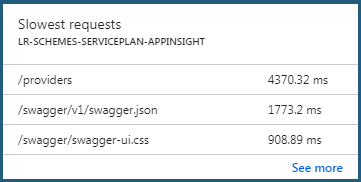




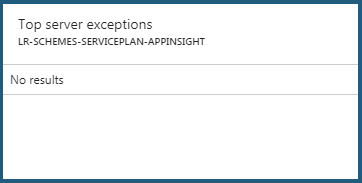
A count of requests that threw uncaught exceptions.



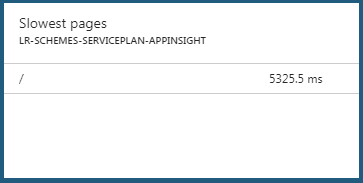
The below one shows which requests might need performance tuning.



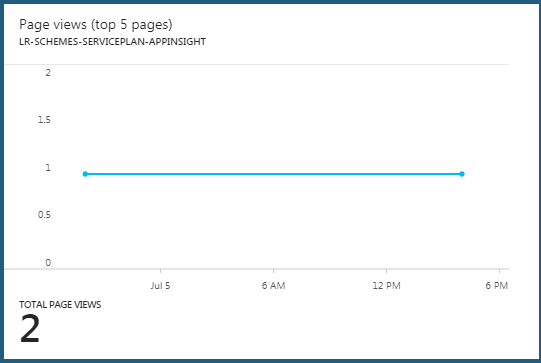
Total of Server exceptions by Problem Id.



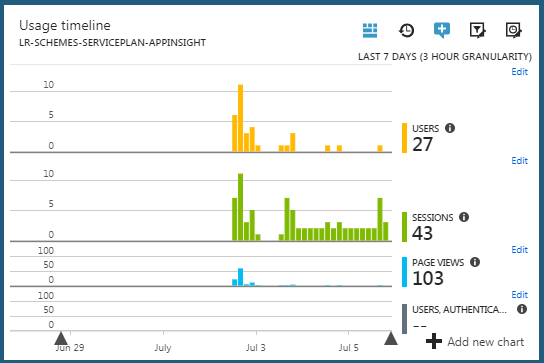
Average of Page view load time by Operation name.



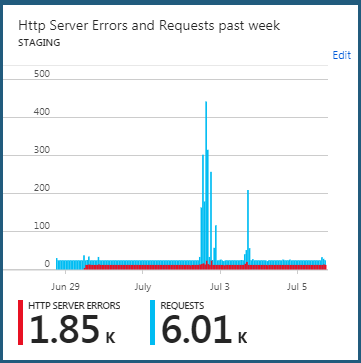
Total of Page views by View page name



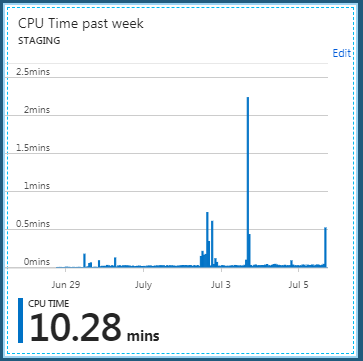
Display metrics usage of application in given point of time. in terms of Users, Sessions, users authentication etc.



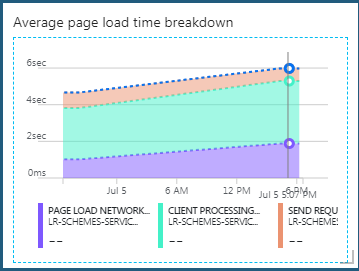
Displays Http servers error request count.



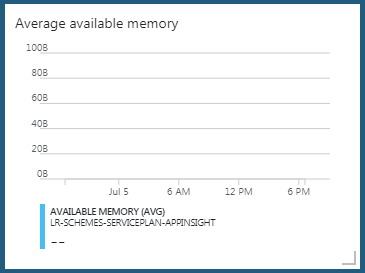
Displays CPU time that application taken in a given time.



Displays the metrics of average page load time.



Displays available memory of server for a given point of time.



Displays metrics of average time taken of processor and process utilization.

