Part 1

50 AZURE ERRORS TROUBLESHOOTING

careerbytecode

careerbytecode.substack.com
+32 - 471408908

1. Virtual Machine (VM) Not Starting

Problem description:
An Azure Virtual Machine (VM) fails to start, staying in a failed or stopped state.

Error:
"Failed to start virtual machine '<VM_Name>'."

What we need to analyze:

- Check if the VM has sufficient allocated resources (CPU, RAM, disk space).
- Review Azure Service Health to check for outages.
- Examine boot diagnostics logs for OS-related errors.
- Check if any recently applied updates caused the failure.

How to troubleshoot:

1. Navigate to **Azure Portal** → **Virtual Machines** → **Boot diagnostics** and check the screenshot/logs.
2. Review **Activity Logs** under **Monitor** to identify if Azure detected a failure.
3. Check if disk corruption exists using **Azure Disk Snapshot** and mounting it to another VM.
4. Validate if the OS disk is running out of space, which can prevent booting.

How to resolve the issue:

- If OS corruption exists, use **Azure Recovery Console** to repair the OS.
- Resize the VM to ensure adequate CPU and RAM.
- Restore from a snapshot if disk corruption is found.
- If a failed extension deployment caused the issue, remove it manually using Azure CLI.

Lessons learned:

- Always enable Azure Backup to restore VMs quickly.
- Regularly check disk utilization to avoid space-related failures.
- Monitor Azure Service Health for potential platform-wide outages.

2. Azure Function App Not Triggering

Problem description:
An Azure Function does not execute when an event occurs.

Error:
"Function host is not running" or no logs in **Monitor**.

What we need to analyze:

- Check if the function is enabled and not in a disabled state.
- Validate that the function's trigger (HTTP, Timer, Blob, etc.) is correctly configured.
- Review Azure Monitor logs for any invocation failures.
- Verify app settings, such as connection strings, are correctly configured.

How to troubleshoot:

1. Navigate to **Azure Portal** → **Function Apps** → **Monitor** to check execution logs.
2. Use **Application Insights** to track function execution failures.
3. If a storage account trigger is used, validate that the connection string is correct.
4. If the function is running on a Consumption Plan, check if it is reaching cold start limits.

How to resolve the issue:

- Restart the function app and monitor logs.
- Increase the Function's plan to a **Premium** or **Dedicated Plan** if cold starts are the issue.
- Recreate and reconfigure missing triggers.
- Update incorrect storage account references.

Lessons learned:

- Always enable **Application Insights** to monitor function execution.
- Avoid using Consumption Plan if the function is critical and time-sensitive.
- Periodically test function execution in staging before deploying.

3. Azure Virtual Network Peering Not Working

Problem description:
Two Azure Virtual Networks (VNets) are peered, but resources in one VNet cannot communicate with those in the other.

Error:
"Request timed out" when trying to connect to resources in the peered VNet.

What we need to analyze:

- Ensure that VNet peering is correctly configured in both directions.
- Verify Network Security Groups (NSGs) are not blocking traffic.
- Check if the virtual machines have correct subnet and IP configurations.
- Confirm whether DNS resolution is working for cross-VNet communication.

How to troubleshoot:

1. Navigate to **Azure Portal** → **Virtual Networks** → **Peering** and verify the peering status.
2. Check NSG rules using **Azure Network Watcher** → **Security Group View** to confirm inbound and outbound rules.
3. Use **Network Watcher** → **Connection Troubleshoot** to check connectivity between resources.
4. Validate that the VMs are using correct DNS settings for name resolution.

How to resolve the issue:

- If peering is misconfigured, delete and recreate it.
- Update NSG rules to allow required traffic across peered VNets.
- If name resolution is failing, configure custom DNS settings.
- Ensure VM firewall rules do not block communication.

Lessons learned:

- Always validate NSG and firewall rules when troubleshooting network issues.
- Enable **Network Watcher** for real-time connection monitoring.
- Regularly test VNet peering with **ping** or **telnet** commands.

4. Azure Storage Account Access Denied

Problem description:
A user or application cannot access an Azure Storage Account despite having permissions.

Error:
"403 Forbidden: This request is not authorized to perform this operation."

What we need to analyze:

- Verify if the Storage Account access key or SAS token is correctly configured.
- Check Azure Role-Based Access Control (RBAC) permissions.
- Ensure Firewall settings on the Storage Account allow access.
- Review Managed Identity configurations if used for authentication.

How to troubleshoot:

1. Navigate to **Azure Portal** → **Storage Accounts** → **Access Control (IAM)** to check permissions.
2. If using SAS tokens, regenerate a new SAS token and test access.
3. In **Firewalls and Virtual Networks**, check if access is restricted by IP.
4. Use **Storage Explorer** to manually test access.

How to resolve the issue:

- Assign the correct **Storage Blob Data Reader/Contributor** role if missing.
- Update firewall settings to allow access from required networks.
- Regenerate the SAS token and ensure the expiry date is valid.
- If using Managed Identity, verify that the identity has proper permissions.

Lessons learned:

- Always check **Storage Account Firewall Settings** before assuming a permission issue.
- Prefer **Managed Identities** over access keys for better security.
- Regularly review IAM permissions to avoid unexpected access failures.

5. Azure Kubernetes Cluster (AKS) Not Scaling Automatically

Problem description:
Azure Kubernetes Service (AKS) does not automatically scale up nodes when pod demand increases.

Error:
"Insufficient CPU/Memory to schedule pod" or "No nodes available."

What we need to analyze:

- Check if the **Cluster Autoscaler** is enabled.
- Ensure node pool settings allow for scaling.
- Review AKS logs for autoscaling errors.
- Validate that quota limits are not preventing node provisioning.

How to troubleshoot:

1. Navigate to **Azure Portal** → **Kubernetes Clusters** → **Node Pools** and check scaling limits.
2. Run `kubectl get nodes` to see if new nodes are being provisioned.
3. Check **Azure Monitor** and **Kubernetes Events** for autoscaling-related errors.
4. Ensure there are available VM quota limits for additional nodes.

How to resolve the issue:

- If autoscaler is disabled, enable it using `az aks update --resource-group <rg> --name <aks-cluster> --enable-cluster-autoscaler`.
- Increase the maximum node count in the node pool settings.
- Request a quota increase if hitting VM limits.
- Restart the **Cluster Autoscaler** if it is stuck.

Lessons learned:

- Always monitor **AKS Cluster Metrics** to anticipate scaling needs.
- Set realistic **min/max node count** for effective autoscaling.
- Periodically test **Cluster Autoscaler** with load tests.

6. Azure App Service Slow Performance

Problem description:
An Azure App Service is running slowly, causing delays in request processing.

Error:
No specific error, but high response times in **Application Insights**.

What we need to analyze:

- Check CPU, memory, and disk utilization.
- Review **App Service Plan** to ensure it has sufficient resources.
- Analyze logs in **Application Insights** for slow request patterns.
- Investigate dependencies (e.g., database or external APIs) that may be causing bottlenecks.

How to troubleshoot:

1. Navigate to **Azure Portal** → **App Services** → **Diagnose and Solve Problems**.
2. Use **Application Insights** → **Performance** to find slow transactions.
3. Run **Azure Monitor** → **Metrics** to check CPU and memory usage.
4. If an external API is involved, test its response time separately.

How to resolve the issue:

- Scale up the App Service Plan if CPU/memory limits are reached.
- Optimize application code and database queries.
- Enable **Azure CDN** to reduce load if static content is slowing down performance.
- Implement **Auto-scaling** to handle traffic spikes.

Lessons learned:

- Regularly monitor **Application Insights** for slow transactions.
- Choose the right **App Service Plan** based on expected workload.
- Implement **caching strategies** to reduce unnecessary load.

7. Azure SQL Database Connection Timeout

Problem description:
An application cannot connect to an Azure SQL Database, experiencing timeouts or delays.

Error:
"Timeout expired. The timeout period elapsed prior to completion of the operation or the server is not responding."

What we need to analyze:

- Check if the database is running and accessible.
- Verify firewall rules and Virtual Network Service Endpoints.
- Review database performance metrics to see if it is under high load.
- Check the application connection string for incorrect settings.

How to troubleshoot:

1. Navigate to **Azure Portal** → **SQL Database** → **Overview** to check status.
2. Check **Azure Monitor** → **Metrics** for DTU (Database Transaction Unit) utilization.
3. Test database connectivity using **SQL Server Management Studio (SSMS)** or `telnet <server> 1433`.
4. Review **Azure Firewall Rules** to ensure the application IP is allowed.

How to resolve the issue:

- If the database is under heavy load, scale up the service tier.
- Update firewall rules to allow application IP access.
- Use **Connection Pooling** to optimize connections.
- If using Private Link, ensure **Private DNS** is correctly configured.

Lessons learned:

- Always enable **Performance Monitoring** to detect high DTU usage.
- Use **Azure SQL Managed Instance** for better networking options.
- Test database failover strategies to minimize downtime.

8. Azure VPN Gateway Not Connecting

Problem description:
A site-to-site VPN connection between an on-premises network and Azure is failing.

Error:
"VPN Tunnel is down" or "IKE authentication credentials are unacceptable."

What we need to analyze:

- Check if the VPN gateway and on-premises firewall configurations match.
- Validate shared key (PSK) settings.
- Ensure the correct routing table is in use.
- Review VPN Gateway logs for connection failures.

How to troubleshoot:

1. Navigate to **Azure Portal** → **VPN Gateway** → **Connections** and check tunnel status.
2. Run `Get-AzVpnGatewayConnection` in **Azure CLI** to check errors.
3. Verify on-premises firewall settings match Azure VPN requirements.
4. Use **Packet Capture** in **Network Watcher** to analyze VPN traffic.

How to resolve the issue:

- If the shared key (PSK) is incorrect, update it in both Azure and on-premises.
- If the VPN tunnel is dropping, enable **IKEv2 KeepAlive** settings.
- If routing issues exist, update **BGP Configuration** if dynamic routing is used.

Lessons learned:

- Always keep a **backup of VPN configuration settings**.
- Regularly test VPN failover to prevent unexpected downtime.
- Use **Azure Virtual WAN** for simpler VPN management at scale.

9. Azure DevOps Pipeline Failing

Problem description:
An Azure DevOps CI/CD pipeline fails during build or deployment.

Error:
"Job Failed" or "Agent unable to reach the target environment."

What we need to analyze:

- Check pipeline logs for build or deployment errors.
- Validate that the build agent is running and has network access.
- Ensure necessary secrets and variables are configured in DevOps.
- Verify that the deployment target (App Service, Kubernetes, etc.) is available.

How to troubleshoot:

1. Navigate to **Azure DevOps** → **Pipelines** → **Runs** and check logs.
2. If using a **self-hosted agent**, ensure it is running and connected.
3. Validate **Azure Service Connection** permissions.
4. Run a manual build to check for missing dependencies.

How to resolve the issue:

- Restart or reconfigure the agent if it is offline.
- Update environment variables and secrets if authentication is failing.
- Increase the build timeout if the pipeline takes longer than expected.
- If using YAML pipelines, validate indentation and syntax.

Lessons learned:

- Use **Azure DevOps Service Connections** with the least privilege.
- Store secrets in **Azure Key Vault** instead of pipeline variables.
- Monitor pipeline execution history for patterns in failures.

10. Azure Load Balancer Not Distributing Traffic

Problem description:
An Azure Load Balancer is not distributing traffic evenly among backend VMs.

Error:
No specific error, but requests are only hitting one instance.

What we need to analyze:

- Check the **Health Probe** status of backend VMs.
- Validate if session persistence settings are causing sticky sessions.
- Review NSG and firewall rules blocking traffic.
- Ensure the Load Balancer has the correct SKU (Basic vs. Standard).

How to troubleshoot:

1. Navigate to **Azure Portal** → **Load Balancer** → **Backend Health** to check VM status.
2. Run `Test-AzNetworkWatcherConnectivity` to simulate traffic flow.
3. Check if session persistence is enabled, which can cause uneven traffic distribution.
4. If using Standard Load Balancer, ensure **Network Security Groups** allow traffic.

How to resolve the issue:

- If backend VMs are unhealthy, restart them and check application logs.
- Adjust session persistence settings for better load distribution.
- If a firewall is blocking traffic, update **NSG Rules**.
- If a Basic SKU is in use, upgrade to **Standard Load Balancer** for better performance.

Lessons learned:

- Always check **Load Balancer Health Probes** when debugging issues.
- Enable **Diagnostics Logs** to track dropped packets.
- Use **Azure Traffic Manager** if global traffic routing is needed.

11. Azure Storage Account Access Denied

Problem description:
Applications or users cannot access an Azure Storage account, resulting in permission errors.

Error:
"403 Forbidden: Authorization Permission Mismatch" or "Authentication failed."

What we need to analyze:

- Check if the Storage Account is accessible from the network.
- Verify if the correct authentication method (Access Keys, SAS, or Managed Identity) is being used.
- Ensure IAM roles and permissions are correctly assigned.
- Check if **Azure Storage Firewall** is blocking access.

How to troubleshoot:

1. Navigate to **Azure Portal** → **Storage Account** → **Networking** and check if public access is restricted.
2. Test access using **Azure Storage Explorer** or `az storage blob list`.
3. Check IAM roles under **Azure Portal** → **Storage Account** → **Access Control (IAM)**.
4. Validate that the correct Shared Access Signature (SAS) token or Access Key is being used.

How to resolve the issue:

- If using Managed Identity, grant the required **Storage Blob Data Reader** role.
- If Access Keys are expired or rotated, update them in the application.
- If Storage Firewall is blocking traffic, allow the necessary IPs.
- If using private endpoints, check **DNS Configuration** and **Private Link settings**.

Lessons learned:

- Always use **Managed Identity** over Access Keys for better security.
- Regularly audit **Storage Access Logs** to track failed access attempts.
- Use **Azure Key Vault** to securely store access credentials.

12. Azure Function Execution Failing

Problem description:
An Azure Function fails to execute or takes too long to complete.

Error:
"Function TimeoutException" or "HTTP 500 Internal Server Error."

What we need to analyze:

- Check **Azure Monitor Logs** for detailed error messages.
- Validate application code for infinite loops or memory leaks.
- Ensure the function app has the right **Plan Type** (Consumption vs. Premium).
- Check if external dependencies (Databases, APIs) are slowing execution.

How to troubleshoot:

1. Navigate to **Azure Portal** → **Function App** → **Monitor** to check logs.
2. Use **Application Insights** to trace function execution times.
3. If function execution takes too long, increase **Function Timeout** in the host.json file.
4. Check the **Scaling Settings** under Function App Plan.

How to resolve the issue:

- If timeout is too low, increase it in the host.json configuration.
- If a database or API call is slow, use **async programming** to optimize execution.
- If the function needs more power, switch from **Consumption Plan to Premium Plan**.

Lessons learned:

- Always set **Application Insights** for tracking function performance.
- Use **Azure Durable Functions** for long-running tasks instead of normal functions.
- Optimize function dependencies to avoid unnecessary delays.

13. Azure Kubernetes Service (AKS) Pod CrashLoopBackOff

Problem description:
A Kubernetes pod in AKS keeps restarting and cannot stay in a running state.

Error:
"CrashLoopBackOff" in `kubectl get pods`.

What we need to analyze:

- Check if the container is crashing due to errors in the application code.
- Review **Resource Limits** and ensure the pod has enough CPU and memory.
- Check if the container image has any missing dependencies.
- Analyze **Pod Logs** to identify the root cause of the crash.

How to troubleshoot:

1. Run `kubectl describe pod <pod-name>` to check pod events.
2. Use `kubectl logs <pod-name>` to view application logs.
3. Check if the **readiness and liveness probes** are misconfigured.
4. If resource constraints are the issue, increase CPU and memory in deployment files.

How to resolve the issue:

- If the pod is failing due to a missing dependency, update the container image.
- If resource limits are too low, increase them in the Kubernetes YAML file.
- If readiness probes are failing, modify the probe thresholds and timeouts.

Lessons learned:

- Always monitor **AKS logs and metrics** for early issue detection.
- Set up **readiness and liveness probes** correctly to avoid unnecessary restarts.
- Use **Azure Monitor for Containers** for real-time visibility into cluster performance.

14. Azure Application Gateway SSL/TLS Issues

Problem description:
Users cannot access an application behind Azure Application Gateway due to SSL/TLS errors.

Error:
"ERR_SSL_PROTOCOL_ERROR" or "Your connection is not private."

What we need to analyze:

- Check if the SSL certificate is valid and correctly configured.
- Ensure the Application Gateway **HTTP settings** use the correct protocol.
- Validate **TLS policy settings** to ensure compatibility with client browsers.
- Confirm that the backend is properly responding to HTTPS requests.

How to troubleshoot:

1. Navigate to **Azure Portal** → **Application Gateway** → **Listeners** to check SSL certificate details.
2. Run `openssl s_client -connect <gateway-ip>:443` to validate SSL/TLS handshake.
3. Use **SSL Labs SSL Test** to analyze the gateway's SSL configuration.
4. Check if **custom TLS policies** are restricting certain protocols (e.g., TLS 1.0).

How to resolve the issue:

- If the certificate has expired, renew and upload a new one to Azure Key Vault.
- If an incorrect protocol is used, adjust the **TLS policy settings** in the Application Gateway.
- If the backend is rejecting HTTPS traffic, ensure SSL termination is correctly configured.

Lessons learned:

- Always track **SSL certificate expiry dates** and set up renewal automation.
- Use **Azure Key Vault** for managing and securing certificates.
- Test SSL/TLS configurations before deploying changes to production.

15. Azure Virtual Machine Disk Space Full

Problem description:
A Virtual Machine (VM) in Azure runs out of disk space, causing services to stop.

Error:
"No space left on device" or "Disk quota exceeded."

What we need to analyze:

- Check the current disk usage on the VM.
- Verify if logs or temporary files are consuming space.
- Identify if the OS or application is writing excessive data.
- Confirm if auto-growth settings are enabled for managed disks.

How to troubleshoot:

1. Connect to the VM via SSH or RDP.
2. Run `df -h` (Linux) or `Get-Volume` (Windows) to check disk usage.
3. Identify large files using `du -sh /*` (Linux) or **Disk Cleanup** (Windows).
4. Check if logs can be compressed or moved to **Azure Storage**.

How to resolve the issue:

- If the OS disk is full, expand the disk from the **Azure Portal** and restart the VM.
- If logs are consuming space, configure **log rotation policies**.
- If using a data disk, attach an additional disk and migrate large files.

Lessons learned:

- Monitor disk space usage with **Azure Monitor and Alerts**.
- Move logs and backups to **Azure Blob Storage** to free up VM space.
- Configure **auto-grow settings** for managed disks to prevent sudden failures.

16. Azure Web App Deployment Failing

Problem description:
Deployment of an Azure Web App is failing due to various reasons, such as configuration issues, missing dependencies, or insufficient permissions.

Error:
"Deployment Failed: ERROR_CONNECTION_TERMINATED" or "HTTP 500 - Internal Server Error."

What we need to analyze:

- Check **Deployment Logs** in the Azure Portal.
- Ensure the correct deployment method is being used (Git, FTP, Zip, or Azure DevOps).
- Validate if the application settings match the environment variables needed.
- Confirm that there are no missing dependencies or package errors.

How to troubleshoot:

1. Go to **Azure Portal** → **Web App** → **Deployment Center** → **Logs**.
2. If deploying via Azure DevOps, check the pipeline logs for failure messages.
3. Check **Application Insights** for backend failures.
4. Use `az webapp log tail` to stream live logs and debug errors.

How to resolve the issue:

- If missing dependencies exist, update the application's dependency file (`requirements.txt`, `package.json`, etc.).
- If using **continuous deployment**, check the Git branch configuration.
- If deployment permissions are insufficient, assign **Contributor** or **Web App Deployment** role to the user or pipeline.

Lessons learned:

- Use **Azure DevOps pipelines** for better CI/CD integration.
- Enable **staging slots** to test deployments before pushing to production.
- Regularly update and test application dependencies.

17. Azure VPN Gateway Connection Issues

Problem description:
An Azure VPN Gateway is unable to establish a connection to on-premises or other virtual networks.

Error:
"Connection Failed: No response from the remote gateway" or "IKE Phase 1 negotiation failed."

What we need to analyze:

- Check if the correct **IPsec/IKE policy** is configured on both ends.
- Verify **shared key (PSK)** and ensure it matches on both ends.
- Ensure firewall rules allow VPN traffic (ports UDP 500 and UDP 4500).
- Review **Azure Network Security Group (NSG) rules** for restrictions.

How to troubleshoot:

1. Run `Get-AzVirtualNetworkGatewayConnection -ResourceGroup <RGName>` in PowerShell to check VPN status.
2. Use **Azure Network Watcher** to diagnose connection issues.
3. Check VPN logs on the on-premises firewall/router for errors.
4. Test the connection using `Test-AzNetworkWatcherIPFlow`.

How to resolve the issue:

- If the shared key is incorrect, reset it using PowerShell or Azure Portal.
- If IKE Phase 1 fails, verify that both ends support the same **encryption algorithm** (AES256, SHA256, DH Group).
- If NSG rules are blocking VPN traffic, update **inbound/outbound rules**.

Lessons learned:

- Always use **Azure Network Watcher** to diagnose VPN issues.
- Monitor VPN connections using **Azure Monitor Alerts**.
- Ensure **both ends** support the same security policies to avoid mismatches.

18. Azure SQL Database Slow Performance

Problem description:
Queries on Azure SQL Database take too long to execute, causing application slowdowns.

Error:
"Query execution timeout exceeded" or "High DTU consumption."

What we need to analyze:

- Check **DTU (Database Transaction Unit) consumption** and CPU usage.
- Identify slow queries using **Query Performance Insights**.
- Check if indexing is missing or fragmented.
- Analyze **throttling issues** caused by resource constraints.

How to troubleshoot:

1. Go to **Azure Portal** → **SQL Database** → **Performance Recommendations**.
2. Use **Query Store** to analyze slow queries.
3. Check `sys.dm_db_index_physical_stats` to see index fragmentation.
4. Monitor DTU usage with `sys.dm_db_resource_stats`.

How to resolve the issue:

- If CPU is high, **scale up** the SQL tier (Basic → Standard → Premium).
- If indexes are fragmented, rebuild them using `ALTER INDEX REBUILD`.
- Optimize queries by **removing unnecessary joins or subqueries**.

Lessons learned:

- Regularly **review query execution plans** to optimize performance.
- Use **Elastic Pools** for better cost optimization across multiple databases.
- Set up **automatic indexing recommendations** in Azure SQL.

19. Azure Load Balancer Backend Health Probes Failing

Problem description:
An Azure Load Balancer is not distributing traffic properly due to failed backend health probes.

Error:
"All backend instances are unhealthy" or "HTTP 503 Service Unavailable."

What we need to analyze:

- Check **Load Balancer Probe settings** and ensure the correct port is configured.
- Verify if backend VMs are **responding to health probes**.
- Ensure NSG rules allow inbound traffic from the Load Balancer.
- Confirm that backend VMs are not overloaded or offline.

How to troubleshoot:

1. Run `Get-AzLoadBalancerBackendAddressPool` to check pool configuration.
2. Use `curl http://<backend-IP>:<probe-port>` to test probe responses.
3. Check NSG rules and allow traffic from **AzureLoadBalancer**.
4. Restart backend VMs and check if they register as healthy.

How to resolve the issue:

- If the probe port is incorrect, update it in the **Load Balancer Health Probe settings**.
- If NSG rules are blocking traffic, allow inbound traffic on the probe port.
- If backend VMs are down, troubleshoot VM health or scale out instances.

Lessons learned:

- Always configure **custom health probes** instead of relying on defaults.
- Monitor **backend instance health** using Azure Monitor.
- Regularly **test backend VM responses** to avoid unexpected failures.

20. Azure Synapse Analytics Query Failing

Problem description:
Queries in Azure Synapse Analytics fail due to resource exhaustion or incorrect configurations.

Error:
"Error: Out of Memory" or "Cannot execute query due to resource constraints."

What we need to analyze:

- Check **Dedicated SQL Pool resource utilization**.
- Identify **data skew issues** affecting distribution.
- Validate **workload management settings**.
- Ensure queries are properly **partitioned** for performance.

How to troubleshoot:

1. Use `sys.dm_pdw_exec_requests` to check running queries.
2. Monitor **Azure Synapse Workload Groups** for excessive resource usage.
3. Run `sys.dm_pdw_nodes_db_partition_stats` to check data distribution.
4. Review **query execution plans** for inefficient joins.

How to resolve the issue:

- If queries are too heavy, increase the **Synapse SQL DWU (Data Warehouse Unit).**
- If data is unevenly distributed, use **HASH DISTRIBUTION** instead of ROUND_ROBIN.
- If workload groups are overutilized, adjust **resource class settings**.

Lessons learned:

- Use **materialized views** to improve query performance.
- Regularly analyze **query execution times** and optimize indexing.
- Monitor **Synapse resource utilization** with Azure Metrics.

21. Azure Kubernetes Service (AKS) Node Pool Scaling Fails

Problem description:
The AKS cluster is unable to scale up or down node pools, leading to resource constraints or unnecessary costs.

Error:
"Failed to scale node pool: Capacity quota exceeded" or "Nodes stuck in NotReady state."

What we need to analyze:

- Check **cluster autoscaler logs** for errors.
- Verify **quota limits** for virtual machine scale sets (VMSS).
- Ensure VM SKU used in the node pool is **available in the region**.
- Look for **resource exhaustion** in the Kubernetes dashboard.

How to troubleshoot:

1. Use `kubectl get nodes` to check node status.
2. Review **Azure Monitor logs** to see scaling failures.
3. Check VMSS limits with `az vm list-usage --location <region>`.
4. If using autoscaler, check events with `kubectl describe cluster-autoscaler`.

How to resolve the issue:

- If VM quota is exceeded, **increase quota in Azure Subscription Limits**.
- If nodes are stuck, restart the kubelet service with `sudo systemctl restart kubelet`.
- If node scaling policies are incorrect, update the AKS **cluster autoscaler settings**.

Lessons learned:

- Regularly **monitor AKS node pool utilization**.
- Enable **autoscaler logs** to detect failures early.
- Always ensure **VM SKU availability** before configuring scaling.

22. Azure Blob Storage Unauthorized Access

Problem description:
Users or applications are unable to access Azure Blob Storage due to authentication or permission issues.

Error:
"403 Forbidden: Authorization permission mismatch" or "Invalid SAS token."

What we need to analyze:

- Verify **RBAC role assignments** for storage accounts.
- Check if the access is via **Managed Identity, SAS token, or Storage Key**.
- Review **network access restrictions** (Private Endpoints, Firewall).
- Ensure **Azure AD authentication** is properly configured.

How to troubleshoot:

1. Check **Access Control (IAM)** in the Azure Portal.
2. Run `az storage blob list` with the appropriate authentication method.
3. Validate **Shared Access Signature (SAS) token expiry**.
4. Test access using Azure Storage Explorer.

How to resolve the issue:

- If using RBAC, assign the correct **Storage Blob Data Contributor** role.
- If using SAS, regenerate the token with correct permissions.
- If blocked by firewall, allow public access or configure **Private Endpoints**.

Lessons learned:

- Always use **Managed Identity** for secure authentication.
- Set **minimum required permissions** for users and apps.
- Monitor **storage access logs** for security breaches.

23. Azure DevOps Pipeline Failing at Build Stage

Problem description:
Azure DevOps pipeline is failing during the build stage, preventing successful deployment.

Error:
"Build Failed: Missing dependencies" or "Agent stopped responding."

What we need to analyze:

- Check if the **build agent has the necessary SDKs** installed.
- Verify if all **dependencies are correctly restored**.
- Ensure the **agent pool is available and responsive**.
- Check for **YAML syntax errors** in the pipeline definition.

How to troubleshoot:

1. Navigate to **Azure DevOps** → **Pipelines** → **Run History** and check logs.
2. If using a self-hosted agent, verify agent status with `az pipelines agent list`.
3. Run `dotnet restore` or `npm install` manually in the build environment.
4. Test pipeline steps locally before pushing changes.

How to resolve the issue:

- If dependencies are missing, add `restore` commands before the build step.
- If the agent is offline, restart the agent service or assign a new agent.
- If using YAML, validate syntax using `az pipelines validate --yaml-file pipeline.yml`.

Lessons learned:

- Always test **pipeline changes locally** before committing.
- Use **Microsoft-hosted agents** for better availability.
- Automate dependency installation in **pre-build steps**.

24. Azure Function Execution Timed Out

Problem description:
An Azure Function is taking too long to execute and fails due to timeout limits.

Error:
"Execution Timeout Expired" or "Function host restart detected."

What we need to analyze:

- Check **execution duration** in Azure Monitor.
- Identify if the function is **consumption-based or premium**.
- Verify if external dependencies (DB, APIs) are causing delays.
- Analyze **cold start delays** if using HTTP-triggered functions.

How to troubleshoot:

1. Go to **Azure Portal** → **Functions** → **Monitor** and check execution times.
2. If using Application Insights, review traces for bottlenecks.
3. Increase the timeout limit if using **Premium or App Service plan**.
4. Test function execution locally using `func start`.

How to resolve the issue:

- If external dependencies are slow, use **async programming**.
- If timeout is reached, switch to **Premium Plan** and increase limit.
- If function cold start is slow, enable **Always On** setting.

Lessons learned:

- Use **durable functions** for long-running workflows.
- Monitor **function execution time** regularly.
- Choose the **right hosting plan** to avoid unexpected failures.

25. Azure Key Vault Secret Retrieval Failing

Problem description:
Applications fail to retrieve secrets from Azure Key Vault, leading to authentication errors.

Error:
"403 Forbidden: Access denied to Key Vault" or "Secret not found."

What we need to analyze:

- Check **RBAC permissions** for the application identity.
- Ensure the **correct secret version** is being accessed.
- Validate **Azure Policy restrictions** on Key Vault access.
- Verify **network access settings** (Public, Private Endpoint).

How to troubleshoot:

1. Run `az keyvault show --name <vault-name>` to check properties.
2. Check logs in **Azure Key Vault Diagnostics**.
3. Use `az keyvault secret show --name <secret-name>` to verify availability.
4. Review **Azure AD permissions** for Managed Identity.

How to resolve the issue:

- If using Managed Identity, assign the **Key Vault Reader** role.
- If secret is missing, check if a **new version** was created.
- If blocked by firewall, configure **Private Endpoints or allow public access**.

Lessons learned:

- Regularly **audit Key Vault access** for security compliance.
- Use **Managed Identity** instead of explicit credentials.
- Monitor **secret retrieval failures** with Azure Monitor.

26. Azure Virtual Machine Boot Failure

Problem description:
An Azure Virtual Machine (VM) fails to boot, leading to service downtime and inaccessibility.

Error:
"Boot diagnostics failed: No OS found" or "VM stuck in Starting state."

What we need to analyze:

- Check if the **OS disk is corrupted** or deleted.
- Review **Azure Activity Logs** for any unauthorized changes.
- Ensure VM **SKU and size are available** in the region.
- Verify **boot diagnostics logs** for failure details.

How to troubleshoot:

1. Navigate to **Azure Portal** → **VM** → **Boot diagnostics** to view screenshots.
2. Check VM status using `az vm get-instance-view --name <vm-name>`.
3. If using a custom image, validate the OS disk integrity.
4. Attach the OS disk to another VM and inspect system logs.

How to resolve the issue:

- If the OS disk is corrupted, **restore from a backup or snapshot**.
- If boot configuration is incorrect, use **Azure Serial Console** to repair GRUB or Boot Manager.
- If VM size is unavailable, resize to a **supported instance type**.

Lessons learned:

- Enable **Azure Backup** for VMs to recover quickly.
- Regularly monitor **boot diagnostics logs**.
- Use **availability zones** for high availability.

27. Azure Load Balancer Not Distributing Traffic

Problem description:
Azure Load Balancer is not routing traffic to backend VMs, causing service disruptions.

Error:
"Backend health probes failed" or "No response from backend pool."

What we need to analyze:

- Check if backend VMs are **running and healthy**.
- Ensure **NSG (Network Security Group) rules** allow traffic.
- Verify **health probe configuration** in the Load Balancer.
- Review **backend pool configuration** for mismatched ports.

How to troubleshoot:

1. Run `az network lb probe show` to check health status.
2. Use `curl` or `telnet` to test connectivity to backend VMs.
3. Check NSG rules with `az network nsg rule list`.
4. Verify **Application Gateway or Traffic Manager** if used in conjunction.

How to resolve the issue:

- If health probes fail, update probe settings to use the correct **port and protocol**.
- If NSG rules block traffic, allow inbound rules on backend VMs.
- If backend VMs are misconfigured, ensure they **respond on the expected ports**.

Lessons learned:

- Always configure **health probes properly** for Load Balancer.
- Review **NSG rules periodically** to prevent accidental blocking.
- Use **Azure Traffic Analytics** to identify routing issues.

28. Azure SQL Database Connection Timeout

Problem description:
Applications are unable to connect to Azure SQL Database, leading to performance issues.

Error:
"SQL Network Error: Connection Timeout Expired" or "Server not found."

What we need to analyze:

- Check if **Azure SQL Server firewall** is blocking connections.
- Verify **database resource utilization** (CPU, DTU, vCores).
- Ensure **application connection string** is correct.
- Check for **regional outages or maintenance**.

How to troubleshoot:

1. Run `telnet <sqlserver>.database.windows.net 1433` to test connectivity.
2. Check **Azure SQL Server firewall rules** in the portal.
3. Use **Azure SQL Query Performance Insights** to detect high load.
4. Run `SELECT * FROM sys.dm_exec_requests` to check for blocking queries.

How to resolve the issue:

- If blocked by the firewall, allow access to client IPs or use **Private Link**.
- If high load is causing timeouts, **scale up SQL tier** or optimize queries.
- If using incorrect credentials, verify **Managed Identity or SQL Authentication**.

Lessons learned:

- Always use **connection pooling** to reduce database stress.
- Enable **automatic scaling** to handle high loads.
- Monitor **SQL performance metrics** with Azure Monitor.

29. Azure Active Directory B2C Sign-In Failing

Problem description:
Users are unable to sign in to an application integrated with Azure AD B2C.

Error:
"User authentication failed: Invalid policy or token."

What we need to analyze:

- Check **user flow policies** in Azure AD B2C.
- Verify **identity provider (IDP) settings** (Google, Facebook, etc.).
- Review **JWT token expiration** and validation errors.
- Check if **API connectors or custom policies** are failing.

How to troubleshoot:

1. Use `jwt.ms` to decode the JWT token and check claims.
2. Check Azure AD B2C logs under **Sign-ins** → **Audit Logs**.
3. Validate **redirect URIs and response types** in the application settings.
4. Run **test user flow** in the Azure portal to debug issues.

How to resolve the issue:

- If policies are misconfigured, update **User Flow or Custom Policy XML**.
- If IDP settings are incorrect, reconfigure **OAuth/OpenID settings**.
- If tokens are expired, increase **token lifetime policy**.

Lessons learned:

- Always test **sign-in flows** in a staging environment.
- Use **Azure AD B2C logs and Application Insights** for debugging.
- Keep **token expiration and refresh strategies** well-defined.

30. Azure Storage Queue Messages Stuck in "Invisible" State

Problem description:
Messages sent to an Azure Storage Queue remain invisible and are not processed by consumers.

Error:
"No new messages available in queue, but messages exist."

What we need to analyze:

- Check if **message visibility timeout** is too long.
- Verify if the **consumer application is running** and retrieving messages.
- Ensure **queue processing logic** is correctly handling retries.
- Look for **poison messages** that repeatedly fail.

How to troubleshoot:

1. Run `az storage message peek` to view messages.
2. Check **queue length and visibility timeout settings** in the portal.
3. Review **Application Insights logs** for processing failures.
4. Run **message dequeue test** with `az storage message get`.

How to resolve the issue:

- If visibility timeout is too high, reduce it in **queue settings**.
- If the consumer app is not running, restart and **debug queue processing**.
- If poison messages exist, implement **dead-letter queue handling**.

Lessons learned:

- Always set **dead-letter queues** for message failure tracking.
- Monitor **queue length and processing time** in Azure Monitor.
- Regularly test **message retrieval and processing logic**.

31. Azure Kubernetes Service (AKS) Node Not Ready

Problem description:
One or more nodes in an Azure Kubernetes Service (AKS) cluster are stuck in a "NotReady" state, affecting workload scheduling.

Error:
`Node <node-name> NotReady`

What we need to analyze:

- Check if the **node is running out of resources (CPU, memory, disk)**.
- Verify **Kubelet logs** to identify connectivity issues.
- Check **Azure VM Scale Set (VMSS) instances** for any failures.
- Ensure **network policies or NSGs are not blocking traffic**.

How to troubleshoot:

1. Run `kubectl get nodes` to list the node status.
2. Check node resource usage with `kubectl describe node <node-name>`.
3. Inspect Kubelet logs: `journalctl -u kubelet -f`.
4. Verify VMSS health in **Azure Portal** → **Virtual Machine Scale Sets**.
5. Check Azure Monitor for **disk space, CPU, or memory pressure alerts**.

How to resolve the issue:

- If the node is out of resources, scale up **AKS node pool** or delete unused pods.
- If Kubelet is unresponsive, restart the node: `az vm restart --name <node-name>`.
- If the issue is network-related, update **NSG rules** to allow AKS traffic.

Lessons learned:

- Set up **Azure Monitor alerts** for node health.
- Use **pod affinity and taints/tolerations** to manage workload distribution.
- Regularly clean up **unused pods, images, and logs** to free up space.

32. Azure Logic App Execution Fails Due to Timeout

Problem description:
An Azure Logic App fails to execute due to long-running operations, leading to workflow failures.

Error:
"Request timeout exceeded" or "Logic App execution failed due to timeout."

What we need to analyze:

- Check if the **action or connector** is taking too long to respond.
- Review **retry policies** and maximum execution time settings.
- Validate **dependent service availability** (e.g., SQL, API calls).

How to troubleshoot:

1. Open **Azure Portal** → **Logic Apps** → **Runs history** to check the failed run.
2. Look at **Action Duration** to see where the delay occurs.
3. If calling an external API, test API response time with `curl` or Postman.
4. If using a database, check for **long-running queries or locks**.

How to resolve the issue:

- Increase the **Logic App execution timeout** in settings.
- If an API call is slow, implement **asynchronous processing** or optimize the request.
- If a database query is causing delays, optimize SQL queries and indexing.

Lessons learned:

- Always implement **error handling and retry policies** in Logic Apps.
- Monitor execution time using **Azure Monitor and Application Insights**.
- Break down **long-running operations** into smaller steps.

33. Azure Key Vault Secrets Not Accessible

Problem description:
Applications cannot access secrets stored in Azure Key Vault, causing authentication failures.

Error:
"403 Forbidden: Access denied to Key Vault" or "Secret not found."

What we need to analyze:

- Check if the **Key Vault access policies** allow the app's identity.
- Verify **Azure Role-Based Access Control (RBAC)** permissions.
- Ensure **Managed Identity (System/User Assigned) is enabled**.

How to troubleshoot:

1. Check Key Vault permissions: `az keyvault show --name <vault-name>`.
2. Validate the application's managed identity using `az identity show`.
3. Use `az keyvault secret show` to test manual secret retrieval.
4. Review **Azure Monitor logs** for access errors.

How to resolve the issue:

- If using RBAC, assign `Key Vault Secrets User` role to the application.
- If using an access policy, explicitly allow **GET and LIST permissions**.
- Ensure **Managed Identity is enabled** and has the correct permissions.

Lessons learned:

- Always use **Managed Identity** for secure access to Key Vault.
- Regularly review **audit logs** to detect unauthorized access.
- Enable **soft delete and purge protection** to prevent accidental deletions.

34. Azure Function App Consuming Too Many Resources

Problem description:
An Azure Function App is consuming excessive memory or CPU, leading to auto-scaling issues or function failures.

Error:
"Function execution exceeded memory limit" or "Too many instances created."

What we need to analyze:

- Check **Application Insights logs** for memory and CPU usage.
- Verify **triggers (Timer, Queue, HTTP) for excessive executions.**
- Review **scaling settings and max instance limits.**

How to troubleshoot:

1. Check execution logs in **Azure Portal** → **Function App** → **Monitor**.
2. If using a Timer Trigger, ensure it **does not overlap executions**.
3. Use `az functionapp plan show` to inspect the hosting plan's limits.
4. If HTTP-triggered, analyze incoming requests to detect spikes.

How to resolve the issue:

- Optimize code to reduce **execution time and memory consumption**.
- Implement **Azure Durable Functions** for long-running workflows.
- Use **Event Grid** to trigger functions asynchronously.
- Scale out to a **dedicated App Service Plan** if needed.

Lessons learned:

- Avoid running **CPU/memory-heavy tasks** inside Azure Functions.
- Monitor function **execution times and failures** proactively.
- Implement **queue-based load leveling** for better scaling.

35. Azure API Management Gateway Not Forwarding Requests

Problem description:
Azure API Management (APIM) gateway is not routing requests to backend APIs, causing service disruptions.

Error:
"502 Bad Gateway" or "Backend service unavailable."

What we need to analyze:

- Check if the **backend API endpoint is reachable**.
- Verify **APIM inbound and outbound policies**.
- Ensure **TLS/SSL certificates are valid**.

How to troubleshoot:

1. Use `curl` or Postman to test the backend API directly.
2. Check **Azure Monitor APIM logs** for request failures.
3. Review **API Policies** for incorrect transformations.
4. Validate **backend service health** using `az network watcher connectivity`.

How to resolve the issue:

- If the backend is down, restart the service and check **availability**.
- If using HTTPS, ensure **TLS/SSL certificates are updated**.
- Modify APIM **policies** if transformations are incorrect.

Lessons learned:

- Always **monitor APIM logs** for backend errors.
- Use **Azure Traffic Manager** for high availability of backend services.
- Automate **certificate renewal** to prevent SSL issues.

36. Azure Storage Account Access Denied from Virtual Machine

Problem description:
A virtual machine (VM) is unable to access an Azure Storage account, causing failures in applications that depend on storage blobs, files, or tables.

Error:
"403 Forbidden: Access Denied" or "This request is not authorized to perform this operation."

What we need to analyze:

- Verify **network access settings** in the Storage Account.
- Check if the **VM has the required identity and permissions.**
- Review **Firewall, VNet, and Private Endpoint configurations**.

How to troubleshoot:

1. Check **Azure Storage Account** → **Networking** settings to ensure VM access is allowed.
2. If using **Managed Identity**, verify it has the `Storage Blob Data Contributor` role.
3. Use `nslookup <storage-account-name>.blob.core.windows.net` to check name resolution.

Test access from VM using:

```
az storage blob list --account-name <storage-account-name> --container-name
<container-name>
```

4. If using a **Private Endpoint**, check DNS resolution and Private Link status.

How to resolve the issue:

- If blocked by a firewall, **allow the VM's public IP or VNet in Storage Account** settings.
- If using **RBAC**, assign `Storage Blob Data Reader/Contributor` role to the VM's identity.
- If using a **Private Endpoint**, update **DNS records** to resolve correctly.

Lessons learned:

- Always use **Managed Identity** for secure storage access.
- Set up **Azure Monitor Alerts** for storage access failures.
- Regularly review **Networking and Firewall settings** in Storage Accounts.

37. Azure App Service Certificate Binding Issues

Problem description:
SSL/TLS certificate is not properly binding to an Azure App Service, causing HTTPS requests to fail.

Error:
"NET::ERR_CERT_COMMON_NAME_INVALID" or "No SSL certificate found for this domain."

What we need to analyze:

- Check if the **certificate is properly uploaded and assigned**.
- Validate **hostname bindings in Azure App Service**.
- Ensure the **certificate is not expired or incorrectly formatted**.

How to troubleshoot:

1. Go to **Azure Portal** → **App Service** → **TLS/SSL Settings** → **Private Certificates** and check the certificate status.
2. Run `openssl s_client -connect <yourdomain>:443` to inspect the certificate.
3. Verify if the **custom domain** is mapped correctly under **App Service** → **Custom Domains**.
4. Check **Key Vault or App Service Managed Certificates** if the certificate is retrieved from there.

How to resolve the issue:

- If using a custom domain, ensure **CNAME or A record is pointing correctly**.
- If the certificate is expired, renew it and **update bindings in App Service**.
- If using **Azure Key Vault Certificates**, verify the Key Vault permissions allow access.

Lessons learned:

- Always enable **Auto-Renewal** for SSL/TLS certificates.
- Use **Azure Application Gateway** for central SSL termination.
- Monitor **certificate expiration** using Azure Policy.

38. Azure Load Balancer Not Distributing Traffic Properly

Problem description:
Traffic is not being evenly distributed across backend VMs or instances behind an Azure Load Balancer.

Error:
"Some backend VMs are not receiving traffic" or "Intermittent connectivity issues to backend services."

What we need to analyze:

- Check **backend health probes** to ensure instances are marked as healthy.
- Verify **load balancing rules and distribution algorithms**.
- Ensure **VMs are in the same VNet and availability set**.

How to troubleshoot:

1. Run `az network lb show --name <lb-name>` to inspect configuration.
2. Use `az network lb probe show --name <probe-name>` to check health probe status.
3. Check **Network Security Groups (NSG) rules** to allow traffic.

Run a **TCP test** on backend VMs to confirm they are responding:

```
nc -zv <backend-vm-ip> 80
```

How to resolve the issue:

- If health probes fail, **fix the backend service** or **adjust the probe settings**.
- If sticky sessions are needed, enable **session persistence** (client IP affinity).
- If traffic is uneven, use **round-robin or hash-based load balancing**.

Lessons learned:

- Always test **backend health probes** after deployment.
- Set up **Azure Monitor Alerts** for load balancer health.
- Regularly check **NSG and firewall settings** for unintended blocks.

39. Azure Virtual Network Peering Not Working

Problem description:
Two Azure Virtual Networks (VNets) are peered but resources cannot communicate across them.

Error:
"Destination Host Unreachable" or "Connection Timeout."

What we need to analyze:

- Verify that **VNet Peering is configured correctly**.
- Check **Network Security Groups (NSG) and Route Tables**.
- Ensure **custom DNS settings are resolving properly**.

How to troubleshoot:

Check the peering status using:

```
az network vnet peering list --resource-group <rg-name> --vnet-name
<vnet-name>
```

1. Test connectivity between resources using `ping` or `Test-NetConnection`.

Check **effective routes** for the VM:

```
az network nic show-effective-route-table --resource-group <rg-name>
--nic-name <nic-name>
```

2. If using **custom DNS**, verify DNS servers are resolving names correctly.

How to resolve the issue:

- If the peering status is **Disconnected**, delete and recreate the peering.
- If NSGs block traffic, allow the required ports and protocols.
- If using **custom route tables**, update them to allow traffic between VNets.

Lessons learned:

- Always test **VNet peering before deployment**.
- Monitor **NSG and Route Tables** for unintended blocks.
- Consider using **Azure Virtual WAN** for better network management.

40. Azure SQL Database Backup Failing

Problem description:
Automated or manual backups of an Azure SQL Database are failing, risking data loss.

Error:
"Backup failed: Unable to access storage account" or "Database size exceeds backup limit."

What we need to analyze:

- Verify **backup retention policies and storage space**.
- Check if **Azure SQL Server has access to the backup storage**.
- Ensure **VNet service endpoints allow SQL and storage access**.

How to troubleshoot:

1. Go to **Azure Portal** → **SQL Database** → **Backups** and check error logs.

Check **long-running transactions** that may delay backups:

```
SELECT * FROM sys.dm_tran_active_transactions;
```

2. If using **Geo-Backups**, ensure replication is not broken.
3. Run `az sql db list-editions` to check if backup size is exceeding limits.

How to resolve the issue:

- If backup storage is full, increase **Storage Account size**.
- If using a **VNet**, enable **Private Link for SQL and Storage**.
- Optimize database size by **purging old data or compressing backups**.

Lessons learned:

- Always configure **long-term retention policies** for backups.
- Enable **Geo-Replication** for high availability.
- Monitor **storage capacity to prevent backup failures**.

41. Azure Virtual Machine Fails to Start

Problem description:
An Azure Virtual Machine (VM) is stuck in the "Starting" state or fails to boot after a restart.

Error:
"Provisioning state: Failed" or "VM is not responding."

What we need to analyze:

- Check if the **underlying disk is corrupted or detached.**
- Verify **availability of compute resources in the selected region.**
- Analyze **Boot diagnostics logs** for errors.

How to troubleshoot:

1. Go to **Azure Portal** → **VM** → **Boot Diagnostics** and check for OS errors.
2. If the VM has a custom image, check for **incompatible drivers.**
3. Run `az vm get-instance-view --name <vm-name>` to check VM health.
4. Try **resizing the VM** to a different SKU and restarting it.

How to resolve the issue:

- If the disk is corrupted, detach it and attach it to another VM for recovery.
- If the VM is using a faulty image, redeploy from a stable version.
- If an **availability zone issue** exists, move the VM to another region.

Lessons learned:

- Always keep a **backup or snapshot** before major VM changes.
- Use **availability sets** or **Azure Site Recovery** for redundancy.
- Regularly monitor **VM health and disk performance.**

42. Azure Functions Timing Out

Problem description:
An Azure Function is taking too long to execute and is hitting the timeout limit.

Error:
"Function execution timeout exceeded"

What we need to analyze:

- Check **function timeout settings** (default: 5 minutes for Consumption Plan).
- Review **code execution time** and optimize it.
- Analyze **dependencies like database calls or API calls.**

How to troubleshoot:

1. Check **Azure Monitor Logs** to find slow execution patterns.

Increase timeout in `host.json` for Premium or Dedicated plans:
json

```json
{
   "functionTimeout": "00:10:00"
}
```

2. Test execution time using **Application Insights**.
3. If using **external APIs**, check for response delays.

How to resolve the issue:

- Optimize **code execution** by reducing redundant operations.
- Move to an **App Service Plan or Premium Plan** if needed.
- Implement **asynchronous processing using Azure Durable Functions.**

Lessons learned:

- Always monitor **execution times** with Application Insights.
- Design functions for **event-driven, small executions.**
- Use **Azure Queue Storage** for workload offloading.

43. Azure Kubernetes Service (AKS) Nodes Not Joining the Cluster

Problem description:
New or existing AKS nodes are not connecting to the cluster, causing workloads to fail.

Error:
"Node NotReady" or "Node failed to join cluster."

What we need to analyze:

- Check **node provisioning status** in AKS.
- Analyze **Kubelet logs** for errors.
- Validate **network and subnet configurations**.

How to troubleshoot:

1. Run `kubectl get nodes` to check node status.
2. Check logs using `kubectl logs -n kube-system <node-name>`.
3. Verify if **VNet, NSG, or firewall rules** are blocking node connectivity.

Restart node using:

```
az vm restart --resource-group <rg-name> --name <vm-name>
```

How to resolve the issue:

- If a node is unhealthy, **delete and recreate it**.
- If the **subnet is full**, expand it or create a new one.
- Update **Kubelet configuration** to allow secure connectivity.

Lessons learned:

- Use **Azure Monitor for Containers** to track node health.
- Regularly check **AKS auto-scaling settings**.
- Keep **Kubernetes versions up to date** to avoid compatibility issues.

44. Azure Application Gateway Backend Pool Unhealthy

Problem description:
Traffic is failing through Azure Application Gateway because backend VMs or services are marked as unhealthy.

Error:
"Backend pool unhealthy: 502 Bad Gateway"

What we need to analyze:

- Check **health probe configuration**.
- Validate **backend service response time and status codes**.
- Analyze **NSG and firewall settings** blocking traffic.

How to troubleshoot:

1. Run `az network application-gateway show-backend-health` to check backend status.
2. Ensure health probes return **HTTP 200** responses.

Test backend service manually using:

```
curl -v http://<backend-ip>:80
```

3. Check **Custom Probe** settings in Application Gateway.

How to resolve the issue:

- If **probes fail**, update the **correct endpoint URL** in health probes.
- If NSGs block traffic, **allow inbound traffic from Application Gateway**.
- Optimize **backend service performance** to respond faster.

Lessons learned:

- Always configure **custom health probes** properly.
- Monitor **backend service performance and scaling**.
- Use **Azure Front Door** for global traffic management.

45. Azure VPN Gateway Connection Drops Frequently

Problem description:
A site-to-site or point-to-site VPN connection to Azure keeps disconnecting.

Error:
"VPN connection lost" or "IKEv2 failure detected."

What we need to analyze:

- Check **VPN Gateway logs** for tunnel failures.
- Analyze **on-prem firewall logs** for blocked traffic.
- Verify **IKE settings and shared keys**.

How to troubleshoot:

Check Azure VPN logs using:

```
Get-AzVirtualNetworkGatewayConnection -ResourceGroupName <rg-name> -Name
<vpn-name>
```

1. Verify **on-prem device logs** for connection failures.

Test continuous connectivity with:

```
ping -t <azure-vpn-ip>
```

2. Ensure the **IPsec/IKE policy matches on both ends**.

How to resolve the issue:

- If a **shared key mismatch** exists, update it in both Azure and on-prem settings.
- If **latency is high**, switch to a **higher SKU VPN Gateway**.
- If using **BGP**, ensure the correct ASN configuration.

Lessons learned:

- Always **document VPN configurations** for easy troubleshooting.
- Use **Azure ExpressRoute** for stable, high-speed connectivity.
- Regularly monitor **VPN logs and performance metrics**.

46. Azure Logic App Triggers Not Firing

Problem description:
An Azure Logic App is not executing its workflow even though the trigger conditions are met.

Error:
"No runs found for trigger" or "Trigger skipped due to condition evaluation."

What we need to analyze:

- Check **trigger history** for failure details.
- Validate **API connections** (e.g., Office 365, SharePoint, or SQL).
- Check **workflow conditions and filtering rules**.

How to troubleshoot:

1. Navigate to **Azure Portal** → **Logic Apps** → **Runs History**.
2. Check if the **API connection is expired** and re-authenticate if necessary.

Validate trigger conditions using:

```
az logic workflow trigger-history show --name <logic-app-name>
--resource-group <rg-name>
```

3. Test the trigger manually using **Run Trigger Now** in the portal.

How to resolve the issue:

- If the **API connection is broken**, reauthorize it.
- If trigger conditions are too strict, adjust the **evaluation logic**.
- If the Logic App is disabled, enable it and re-run the workflow.

Lessons learned:

- Regularly **monitor trigger health** using Azure Monitor.
- Ensure **API connections are active** and refreshed periodically.
- Use **error handling in workflows** to log and retry failed triggers.

47. Azure Resource Lock Preventing Resource Deletion

Problem description:
An administrator is unable to delete or modify an Azure resource due to a resource lock.

Error:
"Failed to delete resource: Resource is locked"

What we need to analyze:

- Check if the **resource has a lock applied** (CanNotDelete or ReadOnly).
- Identify **who applied the lock** and why.
- Validate if dependencies exist that require the lock.

How to troubleshoot:

1. Navigate to **Azure Portal** → **Resource Group** → **Locks** and check for active locks.

Run the following CLI command to list locks:

```
az lock list --resource-group <rg-name>
```

2. Check if an **RBAC policy** is preventing lock removal.
3. Review **audit logs** for lock changes.

How to resolve the issue:

Remove the lock using Azure CLI:

```
az lock delete --name <lock-name> --resource-group <rg-name>
```

-
- If the lock is **needed for protection**, discuss with stakeholders before removal.
- Use **Azure Policy** instead of locks for governance in some cases.

Lessons learned:

- Always document **why a lock was applied** to avoid confusion.
- Use **resource tagging** to indicate locked resources.
- Train teams on using **soft delete and backup features** instead of locks.

48. Azure Automation Runbook Fails to Execute

Problem description:
An Azure Automation Runbook is not executing, failing with authentication or execution errors.

Error:
"Runbook failed: Authentication error" or "Job execution timeout."

What we need to analyze:

- Check **runbook logs** for script execution details.
- Validate **managed identity or service principal permissions**.
- Verify **execution account credentials**.

How to troubleshoot:

1. Go to **Azure Portal** → **Automation Account** → **Jobs** and check the failure reason.
2. If authentication fails, refresh the **service principal credentials**.
3. Test the script manually in the **Azure Cloud Shell**.
4. Check if the **runbook is using outdated PowerShell modules**.

How to resolve the issue:

Update **Azure PowerShell modules** using:

```
Update-Module -Name Az -Force
```

- Ensure the **Managed Identity has correct permissions** in IAM settings.
- If execution times out, **optimize the script** or increase runbook timeout.

Lessons learned:

- Use **Azure Monitor alerts** for failed runbooks.
- Regularly update **PowerShell modules** in the Automation Account.
- Implement **retry logic** in scripts for transient failures.

49. Azure Synapse SQL Pool Failing to Resume

Problem description:
An Azure Synapse SQL Dedicated Pool is not resuming from a paused state.

Error:
"Resume request failed: Pool is in transition state."

What we need to analyze:

- Check **provisioning status** of the SQL Pool.
- Validate **service quota and region availability**.
- Analyze **long-running queries or locks** preventing resume.

How to troubleshoot:

Run the following CLI command to check SQL Pool state:

```
az synapse sql pool show --name <pool-name> --resource-group <rg-name>
```

1. Check **Active Directory permissions** to resume the pool.

Look for **active transactions blocking the pool** using:

```
SELECT * FROM sys.dm_exec_requests
```

2. If the pool is stuck, restart the **Synapse workspace**.

How to resolve the issue:

- If a **quota limit is reached**, request an increase via Azure support.
- If transactions are blocking, **kill the blocking session**.
- Use **automatic scaling policies** to avoid manual resume failures.

Lessons learned:

- Regularly **monitor SQL Pool usage** to prevent unexpected downtime.
- Use **Synapse autoscale** instead of manual pause/resume.
- Ensure **users have correct permissions** for resource actions.

50. Azure DevOps Pipeline Failing Due to Insufficient Permissions

Problem description:
An Azure DevOps CI/CD pipeline is failing due to access or authentication errors.

Error:
"Permission denied: Service connection authentication failed"

What we need to analyze:

- Check **service connections and authentication method**.
- Validate **agent pool permissions** in DevOps.
- Review **OAuth and token expiration settings**.

How to troubleshoot:

1. Navigate to **Azure DevOps** → **Project Settings** → **Service Connections**.
2. Verify if the service principal is **expired or disabled**.
3. Check pipeline logs for **detailed failure reasons**.
4. If using an **agent pool**, ensure the agent has correct IAM permissions.

How to resolve the issue:

- Re-authenticate the service connection or refresh credentials.
- Assign correct **Azure DevOps RBAC roles** to the pipeline identity.

If an agent is failing, restart it using:

```
az pipelines agent restart --pool-name <pool-name>
```

Lessons learned:

- Always set **expiration alerts** for service principal credentials.
- Use **Managed Identity instead of passwords** for authentication.
- Regularly audit **pipeline permissions** for security compliance.

→ TRAININGS

WE ARE DIFFERENT

At CareerByteCode, we redefine training by focusing on real-world, hands-on experience. Unlike traditional learning methods, we provide step-by-step implementation guides, 500+ real-time use cases, and industry-relevant projects across cutting-edge technologies like AWS, Azure, GCP, DevOps, AI, FullStack Development and more.

Our approach goes beyond theoretical knowledge—we offer expert mentorship, helping learners understand how to study effectively, close career gaps, and gain the practical skills that employers value.

16+ Years of operations

91+ Countries worldwide

241 K Happy clients

Our Usecases Platform
https://careerbytecode.substack.com

Our WebShop
https://careerbytecode.shop

CareerByteCode
Learning Made simple

CareerByteCode
All in One Platform

STAY IN TOUCH WITH US!

Website

Our WebShop https://careerbytecode.shop
Our Usecases Platform https://careerbytecode.substack.com

E-mail
careerbytec@gmail.com

HQ address
Belgium, Europe

Social Media
@careerbytecode

Phone
+32 471 40 8908