Project:
Plan, Reduce, Repeat

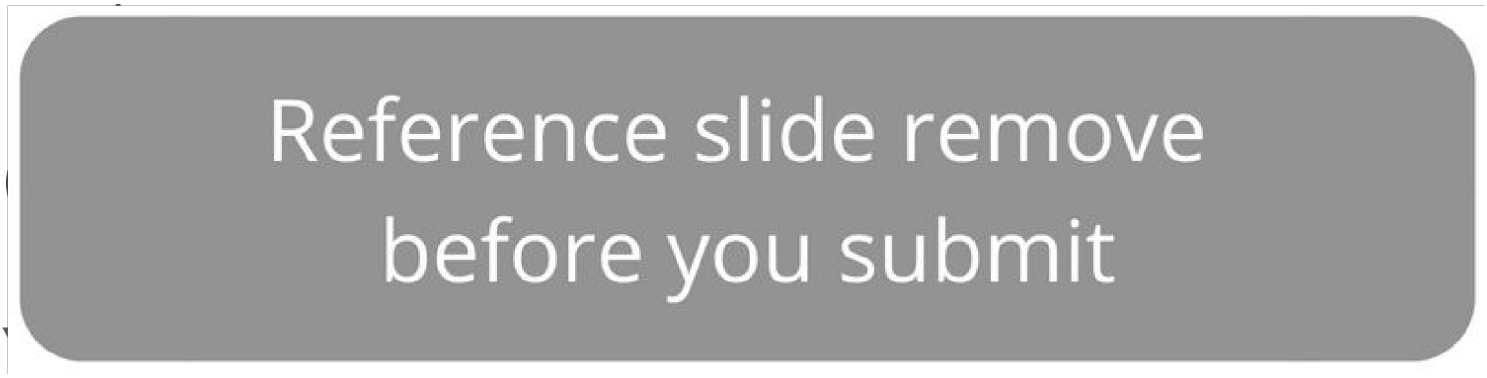
Directions and Submission Template

Nestor Saavedra

4-13-23

How to Use this Template

- Make a copy of this Google Slide deck.
- We have provided these slides as a guide to ensure that you submit all the required components to successfully complete your project.
- When presenting your project, please only think of this as a guide. We encourage you to use creative freedom when making changes as long as the required information is present.
- Feel free to create additional slides if you need more space to write your responses or include screenshots.
- **Feel free to delete this page and the other pages with instructions** before you submit your project.
- **Remember to add your name and the date** to



reseller startup. They already have a small SRE team in place consisting of two other members. You are just finishing up your training period and are now ready to be on your own.

You have a busy week ahead of you as there is a release this week plus your on-call shift. Part of your release duties includes helping to maintain the as-built document by adding this new release, as well as planning for system resource changes. For your on-call shift, you have to respond to alerts as they come in and write up an on-call summary to document your shift. Finally, you'll round out your week by helping to reduce toil. You will have to identify any toil you encounter throughout the week and create a toil reduction plan. After you have a plan all ready, you will need to work on implementing that plan by writing some scripts to help automate tasks.

Scenario 1

Release Day

Release Night

Summary

Tonight is release night, and it will be your first time assisting with a release as an SRE. The process now is manual, with no real consideration for how releases may impact resource allocation. Luckily, your other team members have started implementing an as-built document. You'll have to add tonight's release to the document. The release is a pretty major release with the addition of a new feature that will bring in a large number of new clients. Looking at the results from testing, you can see that this new feature is going to add additional resource requirements as it is both more memory and, to a lesser extent, CPU intensive than before.

Current Release Features

This release will have the following changes that will need to be documented on the as-built design document. The developers have been hard at work implementing the following tickets:

- Ticket 203 added a new catalog for exotic plants. This ticket added new tables in the database to handle the additional catalogs.
- Ticket 202 rearranged the catalog menu in the UI to accommodate the additional catalog, as well as making it more user-friendly.
- Ticket 201 added an additional component to the application, an order processor. The order processor is responsible for batch processing orders on a schedule. The reasoning behind this was to decouple the UI from order processing, and since order processing can be CPU intensive, this decoupling prevents the app from performing poorly. The Design Doc 5247 goes into more detail about the design specifics.
- Ticket 205 fixed a security flaw where attackers could execute a SQL injection attack.

Release Night, cont.

Release Process

The established release process is a manual affair generally done by one of the operations team members. The OPs team generally will download the latest code, shut down the app, run the

database migrations, change or add any needed configurations and then start the app back up. In the past this has caused issues as steps have been forgotten, not all the scripts were executed, the app was not restarted properly, among other issues. During the release window, the OPs engineer would also add new resources as needed. This has led to downtime in the past as the app became overloaded and could not serve requests anymore.

Release Planning

During load testing for this release, it was determined that

- Main Application
  - The new catalog feature increases RAM usages by 25% for the same number of users, while not increasing CPU significantly. Currently, the main application containers are utilizing almost 85% of the RAM allocated.
  - At the current resource allocation, each server can handle 500 concurrent users. Currently, there are 3 application containers to support about 2000 total users, with about 1300 being on at any one time. This release is expected to add about 1.5 to 2.5 times the total number of users, with a need to handle 2600 users concurrently.
- Order Processor
  - This component has a high CPU utilization with moderate RAM requirements. In testing, a full loaded queue used approximately 1 Gb of RAM.
  - The component runs with 2 concurrent processes, pulling orders out of the database and processing them for fulfillment. This component can process 4 orders at a time, with the average order taking between 10 and 15 seconds to complete depending on the size and complexity of the order as well as CPU resource allocation. QA recommends twice the CPU as the main application.
- Database
  - The database was provisioned to handle a much larger application than what the company has now and passed the load tests with flying colors.

As-Built Doc Template

Release Version

Stakeholders

These are the teams and members involved in this reason. This should include ops members, developers, SRE members, database admin, etc

Code Changes

This section should include a list of code changes going into this release separated into groups (for example, by bug fix, feature addition, and security fixes). This should be a short summary of the change with a ticket included to follow up with for more detailed information.

Data and System Changes

This should be formatted similarly to the code changes section, except listing any changes to the data model (database or API changes) or system changes.

Design decision highlights

Document the high-level reasoning behind any design choices. This section should only include a summary of the design decision with links to supporting documentation to follow up with for more detailed information.

Test Section

In this section, list any notable highlights from testing. Things to include here would be any changes to the testing methodology, changes to the test performed, and any tests that are not currently pass (or pass with a warning).

Deployment Notes

Include any changes made to the deployment process or any changes that should be made to improve in feature releases.

As-Built Doc
Release 1

Stakeholders

- Developers
    - John Doe
    - ane Peters
    - Sam Ross
- Ops
    - Jay Smith
- SRE

    - John Robert **Code Changes**

- Security fixes
    - Added new password requirements (Tk-100)
    - Fixed how SQL queries were handled (Tk-103)
- Feature Additions
    - Added new menu options for users (Tk-102)
    - Users can now have middle names (Tk-101)

Data and System Changes

- Data model changes
    - Added columns for middle names in user table (TK-101)
    - Added additional New Menu table (Tk-102)
    - Users table was split into 2 smaller tables (TK-101)

As-Built Doc
Release 1

Design decision highlights

Users table was split into two smaller tables to create more efficient queries and mappings. Keeping it as one big table began to cause slow queries and allowed for a larger number of users. See Design Doc 134 for further discussion.

Test Section

All test suites are passing 100%.

Deployment Notes

The database admins asked for an additional set of scripts to be run for data corrections.

Deployment File Release 1

```
ApiVersion: apps/v1
kind: Deployment
metadata:   name: app-
deployment
namespace: course4
labels:
    app: mainApp
spec:
  replicas: 3
selector:
matchLabels:
app: mainApp
template:
metadata:
labels:
        app: mainApp
spec:
containers:
- name: mainApp        image: nginx:latest
  resources:          requests:
  memory: 256mb            cpu: 250m
  ports:
- containerPort: 80
```

As-Built Doc
Release 2

Stakeholders:

- Developers: John Doe, Jane Peters, Sam Ross
- Ops: Jay Smith
- SRE: John Robert
- Code Changes:

Security fixes:

- Fixed a security flaw where attackers could execute a SQL injection attack (TK-205)

Feature Modification:

- Rearranged the catalog menu in the UI to accommodate the additional catalog (TK-202)

Data and System Changes:

- Data model changes:
    - Added a new catalog for exotic plants (TK-203)
- System changes:

- Added an additional component to the application, an order processor (TK-201)

Design decision highlights:

An order processor was added to the application to batch process orders on a schedule, preventing the app from performing poorly due to the CPU-intensive task of order processing. Refer to Design Doc 5247 for further discussion.

Test Section:

- Tests pass 100%.

Deployment Notes:

Main Application:

- RAM:
  - To accommodate 500 users, 512 MB RAM is necessary [(0.85 x 256 MB) + (0.25 x 256 MB) = 281.6 MB]. Since this is not a valid RAM size, the nearest RAM size of 512 MB should be allocated.
  - CPU: No significant increase is necessary.
- Replicas:
  - Increase the replicas number to 6 because 6 replicas x 500 users = 3000 users > 2600 concurrent users.
- Order Processor:
  - RAM:
    - Allocate 2048 MB RAM, even though load testing showed that the order processor used approximately 1024 MB. This is to avoid 100% RAM utilization.

- CPU:
    - Allocate twice the CPU as the main application (500 MB), as recommended by QA.

- Database:
    - No changes required because the existing database can handle the added functionality without issue.

Deployment File Release 2

```
ApiVersion: apps/v1
kind: Deployment
metadata:
  name: app-deployment
namespace: course4
labels:
    app: mainApp
spec:
  Replicas:          6
selector:
matchLabels:
app:           mainApp
template:
metadata:
labels:
        app: mainApp
spec:
containers:
- name: mainApp
  image: nginx:latest
  resources:
      requests:
          memory: 512mb
          cpu: 250m
          ports:
              containerPort: 80
  name: order_processor
  image: nginx:latest
  resources:
      requests:
          memory: 2048m
      cpu: 500m
  ports:
  -   containerPort: 80
```

Scenario 2

On-Call Shift

On-Call Shift

Summary

Today is your first on-call shift as an SRE. During your shift, you will have to respond to alerts to keep the system running at its best using the on-call best practices learned in this course. During your on-call shift, make sure to be thinking of ways to reduce toil. After your on-call shift is over, you will be responsible for writing a summary of your shift and a post-mortem. On the following slides you will encounter several different “alerts” from your monitoring stack. Each “alert” will contain several different parts that will help you write your on-call log for your shift. Additionally, you’ll encounter an application outage that will require a post-mortem.

Alert Components

Summary -- This will be general knowledge about the systems involved that you would know if you had actually been working at the company. It will include a brief description of the systems involved as well information about how it is managed.

Standard Operating Procedure (SOP) -- This will be a short description of the steps to troubleshoot and potentially correct the cause of the alert.

Log and Monitoring Details -- This section will contain snippets of relevant logs and monitoring data (graphs, metrics, etc.) that are associated with responding to an alert.

On-call Log

After your on-call shift you’ll need to add to the on-call log. There is a provided sample template for you to use that includes all the necessary fields. Remember your on-call log is used to help track recurring alerts/issues as well as providing a record of the steps taken to resolve the issue.

Post-Mortem

Unfortunately there will be an application outage on your shift that will require a post-mortem. You will only be responsible for filling in your involvement, plus you’ll be in charge of creating an action plan and impact assessment.

On-Call Shift -- Alert 1

Order Processing Issues

Summary

You receive an alert that the number of Outstanding Orders is too high. Orders are processed by a separate component from the main application. It runs periodically (every hour currently) to batch process any open orders. Your team has set up some monitors to keep track of how well the order processor is doing.

OP

Number of Outstanding Orders is Too High

If this alert comes through you will need to check the dashboard to see if the Order Processor is overloaded with orders. If there is a high number of orders contact Ops to see if the processor should be run more frequently.

There are logs at /home/sre/course4/order_processing.log. If the server is not overloaded, this is a good place to check for errors. If you encounter any errors, send a message to the developers so that they can troubleshoot.

It is okay to restart this server during business hours. The Order Processor will pick up where it left off after a restart.

On-Call Shift -- Alert 1

Order Processing Issues, cont

Log/Monitoring Details


```
Order Processed
Processing Order 12
Order Processed
Processing Order 13
Order Processed
Processing Order 14
Order Processed
Processing Order 15
Order Processed
Processing Order 16
Order Processed
Processing Order 17
Order Processed
Processing Order 18
Order Processed
Processing Order 19
Order Processed
Processing Order 20
Order Processed
All Orders processed.

Startup...
Starting to process orders.
Processing Order 1
Order Processed
Processing Order 2
Order Processed
Processing Order 3
Order Processed
Processing Order 4
Error Processing Order. Error #12
Error Processing Order. Error #12
Error Processing Order. Error #12
Error Processing Order. Error #12
Error Processing Order. Error #12
Error Processing Order. Error #12
```

On-Call Shift -- Alert 2

Low Storage Alert

ummary

ou receive an alert that the storage is running out on the mount where application logs e being written to. After consulting the SOP, you reach out to the team responsible for

e server. They respond that Steve is normally in charge of handling the logs. Every orning he would run the commands listed in the run book, but he has been out sick for a eek. The other members of the team forgot that it needed to be done, so the mount filled ɔ.

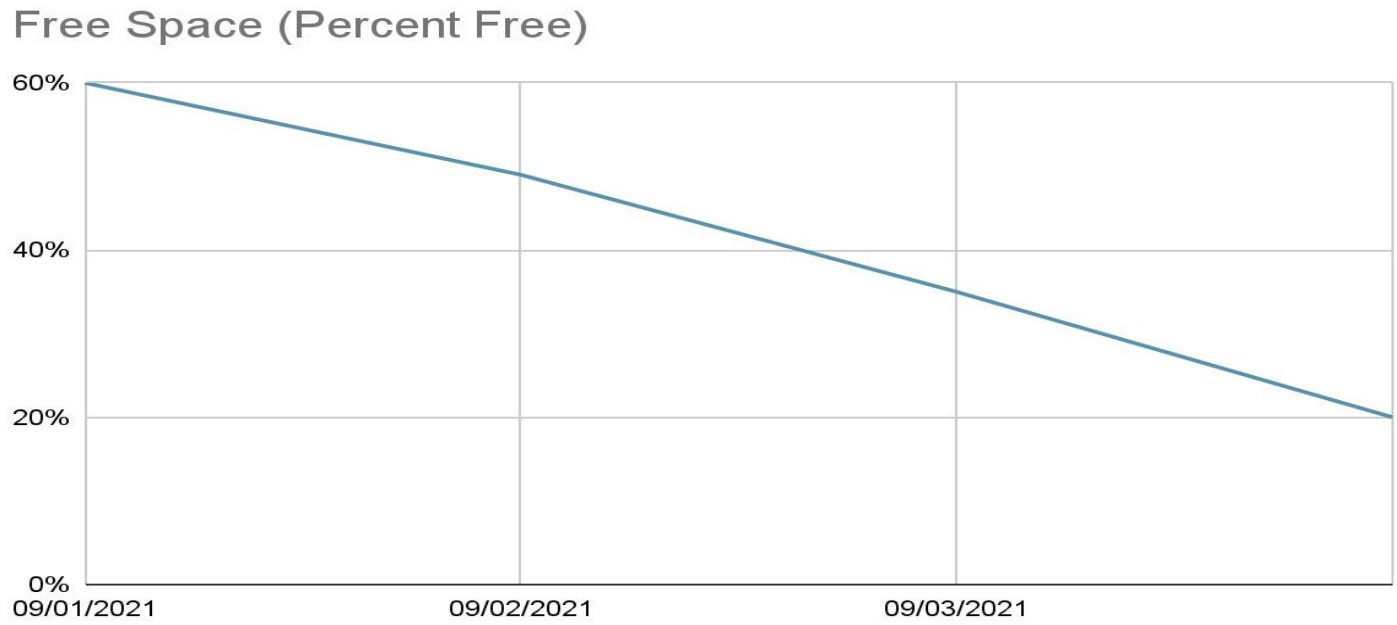
OP

Low Storage

Depending on the specific alert take the following action:

/home/sre/course4/app.log -- If this mount is low on storage, reach out to Compliance. They will know what logs can be cleared out or will request additional storage.

og/Monitoring Details



On-Call Shift -- Alert 3

DNS Troubles

ummary

On-Call Shift -- Alert 3

The networking team recently added a secondary backup DNS server to increase reliability since the one they are using now tends to go down frequently. Your team has several checks in place monitoring the DNS servers to make sure they are up at all times.

OP

DNS Server Not Answering Requests

If you receive this alert, you should check to see if DNS1 or DNS2 is the current server answering requests. After determining which is the active server, check to see if the server is reachable. If the server is not reachable, immediately initiate the failover procedure to prevent any further network disruptions. If the server is reachable, check the logs to determine what the error is. If the active server cannot be brought back online within 5 mins, initiate the failover procedure. Either way, engage the Networking team to bring the standby server back online.

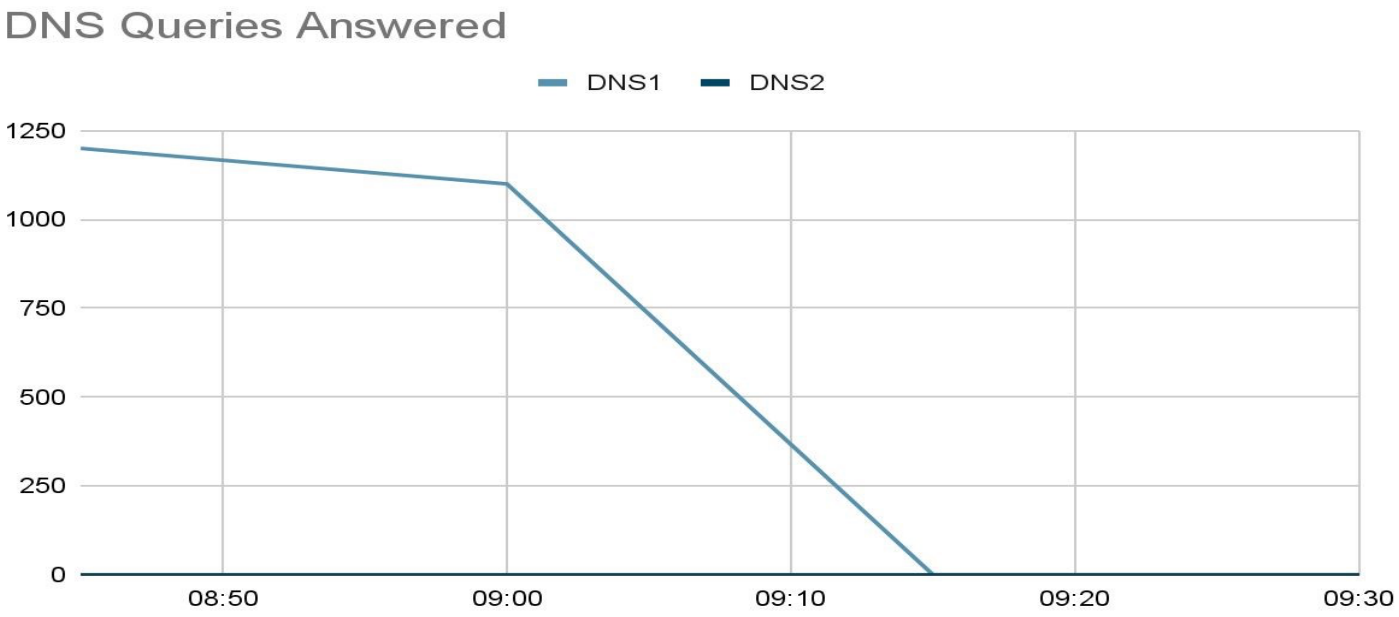
Failover Procedure

1. Determine the active server with the dnsTool.
    a. `dnsTool -q active_server`
2. If the active server is reachable you can initiate the shutdown process. If this command fails, make sure the dns process is shutdown on the server before continuing
    a. `dnsTool -a shutdown -s dns1`
3. Start the failover.
    a. If shutdown was successful:
    `dnsTool -a failover -s dns2`
    b. If shutdown was not successful, include the force flag,
    `dnsTool -a failover -s dns -f`

On-Call Shift -- Alert 3

DNS Troubles, cont

Log/Monitoring Details

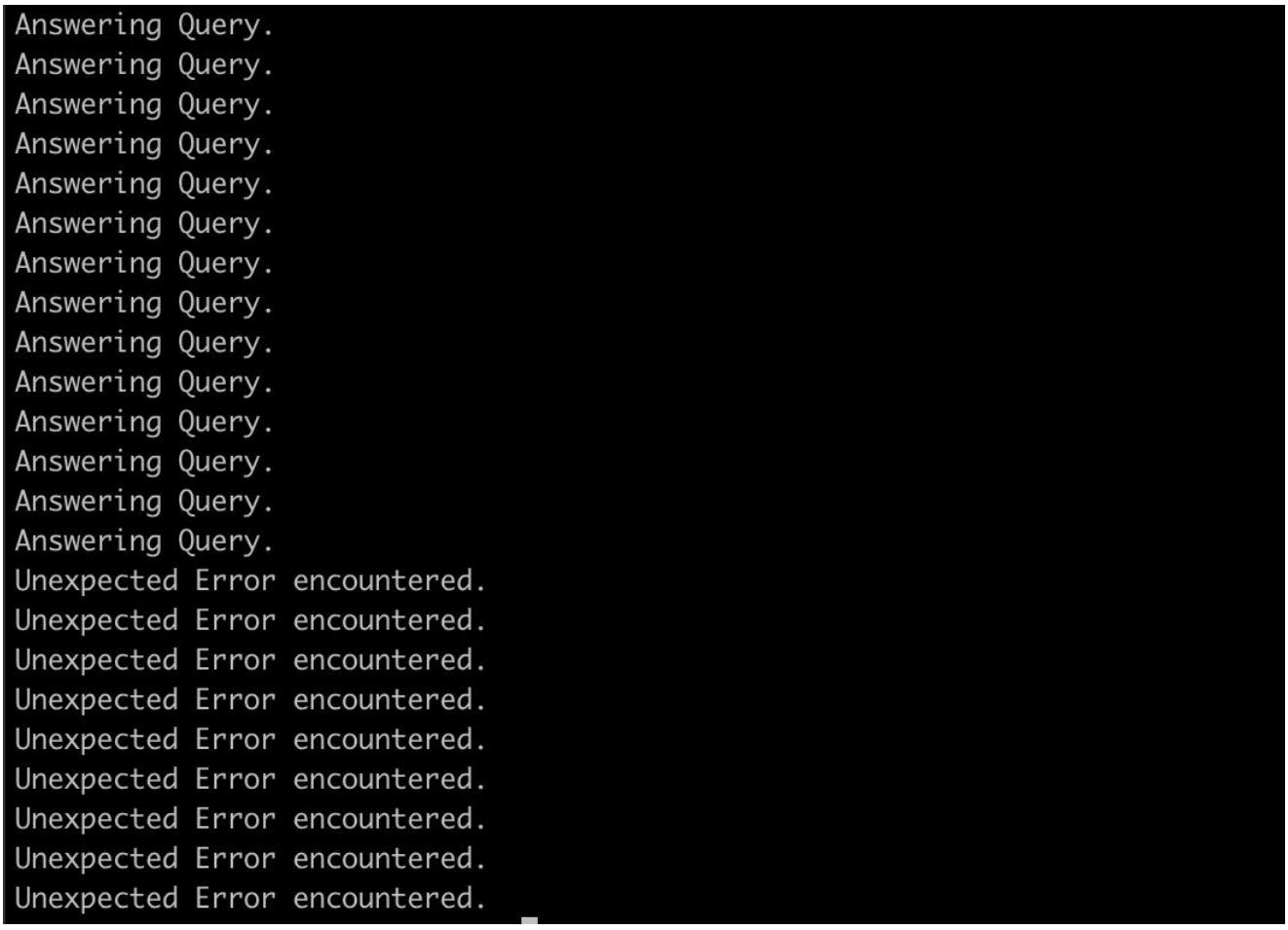


Networking Server Status Page	
Server	Status
DNS1	UP
DNS2	UP

On-Call Shift -- Alert 3

DNS Troubles, cont

og/Monitoring Details



On-Call Shift -- Alert 4

Application Outage

Summary

ou receive the dreaded Application Down alert. Not only do you receive an alert for the application eing down, but Customer Support also sent out a page to get all hands on deck for a report of the pplication being down.

OP

Application Down

If you receive this alert, you need to act immediately. First, verify the application is indeed unreachable. If the application is unreachable, check to make sure the hosts are up and the application processes are running. You must start escalation for this immediately after verification the app is unreachable. Contact the following POCs:

Customer Support -- Susan Vega

Networking -- Bob Sparrow

Ops -- Glen Hammer

Database Admin -- Karen House

Development Team – Gal Tree

og/Monitoring Details

Main App Status	
Endpoint or Host	Status
exoticplant.plant	UNREACHABLE
planthost1.internal	UP
planthost2.internal	UP
exoticplant.plant.internal	UNREACHABLE

On-Call Shift -- Alert 4

Application Outage, cont

.og/Monitoring, cont.

```
Info: Processing Request 407
Warming: Timeout. Retrying 428
Info: Processing Request 439
Warming: Timeout. Retrying 447
Warming: Timeout. Retrying 941
Warming: Timeout. Retrying 168
Warming: Timeout. Retrying 205
Warming: Timeout. Retrying 278
Info: Processing Request 439
Info: Placing Order 492
Warming: Timeout. Retrying 814
Info: Placing Order 520
Warming: Timeout. Retrying 662
Info: Processing Request 776
Info: Processing Request 548
Info: Processing Request 559
Warming: Timeout. Retrying 905
Info: Placing Order 948
Info: Placing Order 340
Error: Var is 10 RETRYING
```

Application Outage, cont

og/Monitoring, cont.

On-Call Shift -- Alert 4

09:15 Hey we have reports of an application outage and we can not reach the app either. **FROM: svega**
09:16 I have an alert for that too. I'm looking at things now, will start a communication channel to coordinate. Checking logs and app servers now. **FROM: YOU**

09:20 -- !svega !bsparrow !ghammer !khouse !gtree we have an application outage **FROM:**
YOU.
0930 -- Everything looks good from the network **FROM: sparrow**
0932 -- I can access the DB and it is reporting back normal **FROM: khouse**
0935 -- Everything here looks normal. FROM: ghammer
0937 -- We are still reviewing logs and seeing if we can reproduce on our end FROM: gtree
0938 -- We should try restarting the app, Maybe that will help FROM: ghammer
0940 -- Maybe that will help. FROM: svega
0943 -- Okay I will try. Bringing down. FROM: YOU
0945 -- App is down. Bring back up. FROM: YOU
0947 -- App is starting. FROM: YOU
0952 -- Main app is back up. FROM: hammer
0955 -- App is still not respond. FROM: svega
0956 -- I'm sending you some new logs !gtree these look off FROM: hammer
1005 -- !sre !ghammer when was the last deploy? What were the details? This looks like a qa build. FROM: gtree
1007 -- I did a deploy with one of the devs to qa to do some testing. Let me check. FROM: ghammer
1010 -- I think there was a mixup when doing the deployment. The wrong scripts was used and that build was deployed to prod. FROM hammer
1011 -- Were there any migrations for that !ghammer FROM: khouse
1012 -- No, just code changes. FROM: hammer
1013 -- Thats good. We should be able to just revert back then. !svega
1015 -- Let me take down the app and redeploy it. FROM: YOU

On-Call Shift -- Alert 4

1017 -- App is down. Bring back up. FROM: YOU
1023 -- App is starting. FROM: YOU
1026 -- Main app is back up. FROM: hammer
1030 -- Everything looks like it is responding now. FROM: svega

On-Call Summary Log Template

Date/Time: 10:00 -- Alert 1: Order Processing Issues

Troubleshooting

I received an alert indicating that the number of outstanding orders was too high. Our orders are processed by a separate component from the main application, and it runs periodically, every hour currently, to batch process any open orders. My team has set up some monitors to keep track of how well the order processor is doing.

To troubleshoot this issue, I followed our standard operating procedure (SOP):

1. I observed the Orders Dashboard Graph and noticed that the number of outstanding orders had increased sharply from 3 at 9:00 am to 15 at 10:00 am. Additionally, the number of orders processed had decreased from 16 at 9:00 am to 0 at 10:00 am. I then moved on to the next step in the SOP.

2. I checked the register folder at the address /home/sre/course4/order_processing.log. As the server was not overloaded, I verified that an error of the type "Error Processing Order. Error #12" was observed at 10:00 am batch. I sent a message to the developers to fix the problem.

3. During business hours, the server restarted, and the Order Processor picked up where it left off after the restart.

Resolution

To solve the alert of too high outstanding orders numbers, I took the following steps:

1. I sent a message to the developer group to fix the problem of the observed errors, such as "Error Processing Order. Error #20."

2. The server restarted during business hours, and the Order Processor picked up where it left off after the restart.

Alert 2: Low Storage Alert at 09.03.2021/08:00

Troubleshooting:

When I received the alert about low storage on the mount where application logs are being written to, I immediately consulted the SOP to determine the next steps. Following the protocol, I reached out to the team responsible for the server to find out what was going on. I learned that Steve, who oversaw handling the logs, had been out sick for a week. Normally, he would run the commands listed in the run book every morning, but the other members of the team had forgotten to do so, leading to the mount filling up.

To further investigate, I checked the free space from 9 January 2021 until 9 April 2021 and observed a reduction in the free space by around 40%. As of 09/03/2021, we only had 35% of free space remaining.

Following the SOP for low storage, I proceeded to the next step:

1. I went to /home/sre/course4/app.log and realized that the mount was low on storage. As per the protocol.

2. I reached out to Compliance to determine what logs could be cleared out and requested additional storage.

Resolution:

After reaching out to Compliance, they provided the necessary support to clear out logs and allocate additional storage space. This helped to resolve the low storage alert, ensuring that there was sufficient storage space available for the application logs to be written to.

Moving forward, we have put measures in place to ensure that routine maintenance tasks are performed regularly and diligently, even in the absence of key team members.

Date/09:15 -- *Alert 3.* DNS Troubles Troubleshooting

Our networking team recently added a secondary backup DNS server to increase reliability, as the primary server tends to go down frequently. To ensure that the DNS servers are always up, I worked with my team to put several monitoring checks in place.

Issue: At 9:15, I received an alert that DNS1 was not responding to requests. To troubleshoot the issue, I followed the DNS Server Not Answering Request SOP, which involved the following steps:

1. **Check Server Status**: I first checked that both DNS1 and DNS2 were up and reachable by looking at the networking server status page. I confirmed that both servers were up.

2. **Check Logs:** Since both servers were up, I then checked the logs to determine the error. I found that DNS1 had encountered an "Unexpected Error." After trying to bring the active server back online for 5 minutes without success, I initiated the failover procedure.

3. **Initiate Failover Procedure:** I followed the failover procedure as per the SOP, which involved the following steps:

    - Determine Active Server: I used dnsTool to determine the active server: dnsTool -q active_server

    - Shutdown Active Server: If the active server is reachable, I initiated the shutdown process with this command: dnsTool -a shutdown -s dns1. If the shutdown command failed, I made sure that the dns process was shutdown on the server before continuing.

    - Start Failover: I started the failover with this command: dnsTool -a failover -s dns2. If the shutdown was unsuccessful, I included the force flag: dnsTool -a failover -s dns -f.

Resolution:

Following the failover procedure ensured that the DNS services continued without disruption. I followed the procedure because DNS1 had encountered an "Unexpected Error," and the status of both DNS servers was UP. After trying to bring the active server back online for 5 minutes without success, the failover procedure was the best course of action to maintain DNS functionality.

Date/09:15: Alert 4. Application Outage Troubleshooting:

At 9:05 AM, an Application Down alert was received from Susan Vega at Customer Support, indicating that the application was down. An all-hands-on-deck page was sent out to report the issue.

1. The SRE team followed the SOP Application Down procedure:

- Verified that the application was indeed unreachable by checking the monitoring details of the endpoint and host status. The exotic pant.plat and exotic plant.plant.internal were both unreachable.

- Verified the logs, which showed the following fail: “Error: Var is 10 retrying”. Requests were also in the state: “Warming:Timeout. Retrying ”.

2. After verifying that the application was down, the SRE team immediately initiated the escalation process and contacted the following POCs:

    - Customer Support: Susan Vega

    - Networking: Bob Sparrow

    - Ops: Glen Hammer

    - Database Admin: Karen House

    - Development Team: Gal Tree

3. Two attempts were made to recover the service:

    - The first attempt involved restarting the application by the SRE team, but it was still unreachable, as verified by Customer Support.

    - The second attempt involved a conversation between Ops and Database Admin at 10:11 AM, where Ops asked Database Admin if there were any migrations. Ops was informed at 10:12 AM that there were some changes in the code.

    - At 10:15 AM, the SRE team decided to take down and redeploy the app. Customer support confirmed that the app was reachable again at 10:30 AM.

Resolution:

- The root cause of the issue was identified as changes made to the application code by the Ops team.
- The SRE team resolved the issue by taking down the application and redeploying the code, resulting in a successful resolution of the outage.

Post-Mortem Template

Incident Title -- Date/Time Stakeholders

This should include all teams and individuals who were involved in the incident.

Incident Timeline

This is a timeline of the events from when the incident was reported to resolution. Make sure to include both events by actual persons (Joe logged on to server1 and restarted the service) as well as system events (From the logs in server2, we see that network connectivity stopped at 14:32).

Impact

This section should include an impact assessment that describes how the business, customers, and systems were affected. The outage affected the order processing system preventing orders from being processed. This led to customers having delayed orders, as well as having to pull additional business resources in to process orders manually. This led to a loss of revenue for the business.

Resolution

Describe what was specifically was done to resolve the issue. This section can be used to document scripts, commands, actions, or vendor support engaged that may be useful for follow-up or automation.

Action Plan

This will be a plan of action to prevent the incident from reoccurring. It should include any safeguards to be implemented, automation to be performed, additional redundancy to be added, etc. This should also include a breakdown of who will perform what and when it should be implemented by.

Post-Mortem

Application Outage -- Date/Time

Stakeholders

Incident Timeline

Impact

Resolution

Action Plan

Application Outage -- Date/09:15

Stakeholders:

- Customer Support
- Susan Vega
- Networking
- Bob Sparrow
- Ops
- Glenn Hammer
- Database Admin
- Karen House
- Development Team

- Gal Tree
- SRE
- Nestor Saavedra

Incident Timeline:

- 09:15: Susan Vega from Customer Support reports an application outage. “Hey we have reports of an application outage and we can not reach the app either. **FROM:** svega”
- 09:16: Nestor Saavedra, a member of the SRE Team, checks the system monitoring to ensure the system is running smoothly. Regarding any issues, he follows standard operating procedures to coordinate with other relevant groups for quick resolution. Susan Vega (from Customer Support), Bob Sparrow (from Networking), Glenn Hammer (from Ops), Karen House (from Database Admin) and Nestor Saavedra (from SRE). “I have an alert for that too. I’m looking at things now, will start a communication channel to coordinate. Checking logs and app servers now. **FROM: YOU”**
- 09:20 -- !svega !bsparrow !ghammer !khouse !gtree we have an application outage **FROM: YOU**.
- 09:30: Bob Sparrow reported the network is working. “Everything looks good from the network **FROM: sparrow”**
- 09:32: Karen House was able to access the database and it is reporting back normal. **FROM: khouse**
- 09:35: Glenn Hammer reported that the operations are normal. “Everything here looks normal. FROM: ghammer”

- 09:37: Gal Tree reports that they are still reviewing logs and seeing if they can reproduce the issue. “We are still reviewing logs and seeing if we can reproduce on our end FROM: gtree”
- 09:38: Glenn Hammer suggests to restart the application. “We should try restarting the app, Maybe that will help FROM: ghammer”
- 09:40: Susan Vega agrees to restart the application. “Maybe that will help. FROM: svega”
- Between 09:43 to 09:47, Nestor Saavedra proceeds with the restart process.
- 09:52: Glenn Hammer reports that the main app is back up.
- 09:55: Susan Vega reports that the main application is still unreachable.
- 09:56: Glenn Hammer proceeded with sending new logs to Gal Tree. “I’m sending you some new logs !gtree these look off FROM: hammer”
- 10:05: Gal Tree inquires with Nestor Saavedra and Glenn Hammer from the SRE team to gather information regarding the details of the last deployment, while also pointing out that it appears to be a QA build.
- 10:07: Glenn Hammer explains that he recently conducted a deployment with one of the developers to the QA environment in order to perform some testing. “I did a deploy with one of the devs to qa to do some testing. Let me check. FROM: ghammer”
- 10:10: Glenn Hammer indicates that there may have been a mix-up during the deployment process resulting in the unintended deployment of the build to production. “I think there was a mixup when doing the deployment. The wrong scripts were used and that build was deployed to prod. FROM hammer”

- 10:11 - Karen House inquires Glenn Hammer if there were any migrations for the application. She uses the tag "!ghammer" to ensure that he receives her message. “Were there any migrations for that !ghammer FROM: khouse”
- 10:12 - Glenn Hammer replies that there were no migrations, only code changes. "FROM: hammer".
- 10:13 - Susan Vega expresses her delight with the news, saying "That's good. We should be able to just revert back then. “Thats good. We should be able to just revert back then. !svega”
- Between 10:15 and 10:23, Nestor Saavedra takes down the application, redeploy it, and then start it again. He reports the steps he is taking and identifies himself as the sender with the tag "FROM: YOU".
- 10:26 - Glenn Hammer confirms that the main application is back up. “Main app is back up. FROM: hammer”
- 10:30 - Susan Vega confirms that the main application is up and responding. “Everything looks like it is responding now. FROM: svega”

Impact

- Business: The outage lasted for 1 hour and 15 minutes, during which the web was unable to process orders, resulting in a loss of revenue for the business.
- Customer: During the outage, customers were unable to purchase items through the e-commerce system. Additionally, the outage damaged the e-commerce's reputation, causing customers to potentially avoid making purchases and not recommend the site to others.
- System: The order processing system was affected, preventing orders from being processed.

Resolution:

- The Ops team had made code changes to the application, and the SRE team resolved the issue by taking down the application and redeploying the code.

Action Plan: To prevent future outages, the following action plan could be implemented:

Plan 1:

1. Migrate the service to the cloud: This is because moving to the cloud offers many advantages such as cost savings, improved security, flexibility, better quality control, disaster recovery, loss prevention, and automatic software updates. By migrating to the cloud, we can leverage these benefits and reduce the risk of future outages.
2. Add in the system architecture the following components and devices: A. Deploy the architecture in a Cloud Region: By deploying the architecture in a Cloud Region, we can improve the application's availability and reduce the risk of regional outages. B. Deploy every component in a VPC (Virtual Private Cloud): This provides isolation and improves the security of the system. C. Load Balancer to share the load to different virtual machines: This distributes the load across different machines and ensures that the application can handle large amounts of traffic. D. Virtual Machines that include a Kubernetes cluster with micro-services for the main application and order processor: By using micro-services and Kubernetes, we can improve the scalability and reliability of the application. E. Implement an auto-scaling group: This enables the application to scale out or scale in based on the load of the server, which ensures that the application can handle sudden spikes in traffic and reduces the cost of running the application during low traffic periods. F. Add a Primary

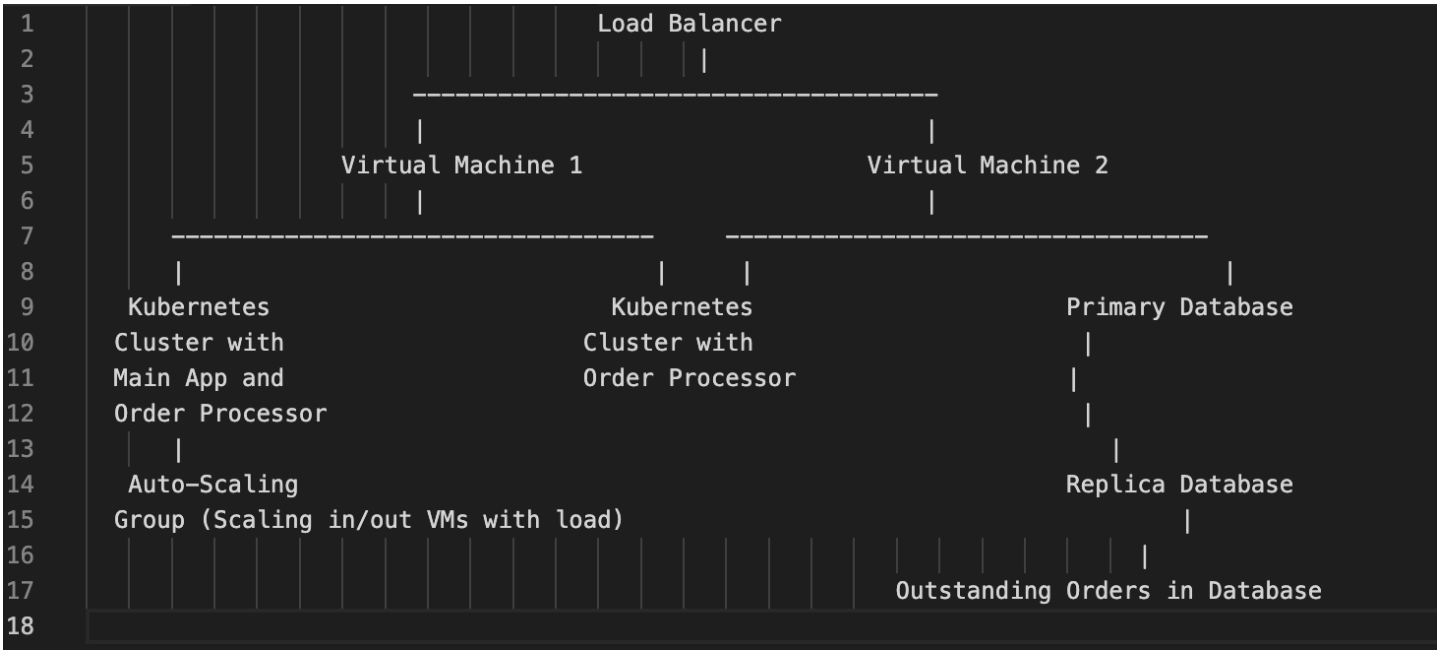
Database in an availability zone and another Replica Database in another availability zone: This ensures that the system can handle a sudden increase in orders and reduces the risk of data loss in case of an outage in one availability zone.

3. Implement a Disaster Recovery Plan with the goal of having zero downtime: The pilot light DR strategy involves replicating all infrastructure in another region, but in this case, only a Replica Database is required. This ensures that the system can recover quickly in case of a disaster and ensures that the business can continue to operate without interruption.

4. Automate all tasks to reduce toil: A. Use Terraform software to automate the deployment of all components of the architecture: This ensures that the deployment process is reliable, repeatable, and faster. B. Using run book with the scripts to deploy the software: This enables the team to automate repetitive tasks and reduce the risk of errors during deployments.

To avoid another change in code causing an outage in the application, it is necessary to test the code in an environment like our production environment. We can use different strategies to deploy the new version of the code to avoid the outage, such as Rolling Deployment, Canary Deployment, and Blue-Green Deployment. These strategies ensure that the new version of the code is gradually rolled out to the system and any issues can be caught before the entire system is affected.

DR strategy:



Plan 2: On-Premise Infrastructure Deployment

Objective: Deploy a scalable on-premise infrastructure to handle the load of the main application and order processor.

Components:

1. **System Architecture:**

A. Load Balancer: - To distribute the load to different Virtual Machines with the main application and order processor.

B. Virtual Machines: - Including a Kubernetes cluster with micro-services for the main application and order processor.

C. Auto-scaling Group: - Implement an auto-scaling group to adjust the number of virtual machines with the server's load.

D. Databases: - Add a primary database and a replica database to handle requests of the order processor in the main database with outstanding orders.

2. **Automation**:
    - Automate all tasks to reduce toil.

A. Terraform Software: - Use Terraform software to automate the deployment of all components of architecture.

B. Run Books: - Use run books with scripts to deploy the software.

3. **Deployment Strategies:**
    - Implement different deployment strategies to avoid outages during code updates.

A. Rolling Deployment: - Deploy updates gradually, one server at a time.

B. Canary Deployment: - Test the new version on a small group of servers before deploying it to the entire infrastructure.

C. Blue-Green Deployment: - Create a separate environment for the new version and switch to it when ready.

Scenario 3

Toil Reduction

Toil Reduction Plan

Summary

Now that you have spent some time on your own as an SRE, you now have to round out your week by handling some of the toil you encountered. Looking through the on-call summary, post-mortem, as-built design doc, and your experience, you decided that there are several ways to reduce toil. You start by listing out 5 of the major items for this week. For each one, you analyze the impact on the business and what you gain by automating the task. After that, you will need to implement three of these items in pseudocode to help your team move forward.

Current Toil Items

1. **Deployment Process**:

   Manually deploying code changes is considered toil because it is a repetitive and error-prone task that takes up a significant amount of time. Automating the deployment process can eliminate the need for manual intervention, reduce the risk of errors, and speed up the deployment process.

Steps:

1. Evaluate the current deployment process to identify pain points and areas that can be automated.
2. Implement a deployment pipeline using a tool such as Jenkins or CircleCI that automates the build, testing, and deployment of code changes.
3. Integrate automated testing and quality assurance checks into the deployment pipeline to catch errors early in the process.
4. Configure the pipeline to automatically deploy changes to staging or production environments based on predefined rules or triggers.

5. Monitor the pipeline for failures and adjust as needed to improve the reliability and efficiency of the deployment process.

Justification for automation:

6. Eliminates the need for manual intervention.
7. Reduces the risk of errors.
8. Speeds up the deployment process.
9. Improves reliability and efficiency.

2. **Password Reset:**

   Manually resetting user passwords is considered toil because it involves a lot of manual work, such as verifying user identity, generating new passwords, and communicating the new credentials to the user. Automating the password reset process can reduce the workload on the SRE team, improve security by enforcing password policies, and enhance user experience by providing a self-service option.

Steps:

1. Develop a password reset workflow that integrates with user identity verification tools and password policy enforcement tools.
2. Implement a self-service option that allows users to reset their passwords securely without SRE intervention.
3. Integrate the password reset workflow with monitoring and alerting tools to detect and prevent fraudulent password reset attempts.

Justification for automation:

4. Reduces the workload on the SRE team.
5. Improves security by enforcing password policies.
6. Enhances user experience by providing a self-service option.

3. **System Monitoring and Alerting:**

   Manually monitoring and alerting of system metrics is considered toil because it involves constantly checking system metrics for anomalies and generating alerts manually. Automating the monitoring and alerting process can improve system reliability by providing real-time insights into system health, reduce the workload on the SRE team, and enable proactive resolution of issues.

Steps:

1. Develop a system monitoring strategy that defines the metrics to be monitored, alert thresholds, and escalation procedures.
2. Implement a monitoring tool such as Nagios or Zabbix to automatically monitor system metrics and generate alerts based on predefined rules.
3. Integrate the monitoring tool with incident management tools such as PagerDuty or OpsGenie to enable proactive resolution of issues.

Justification for automation:

4. Improves system reliability by providing real-time insights into system health.
5. Reduces the workload on the SRE team.
6. Enables proactive resolution of issues.

4. **Infrastructure Creation:**

   Manually creating infrastructure resources is considered toil because it involves repetitive tasks, such as creating VMs, databases, and load balancers, and configuring them manually. Automating the infrastructure creation process can reduce the workload on the SRE team, improve consistency and reliability, and enable faster provisioning of resources.

Steps:

1. Develop an infrastructure creation workflow that defines the infrastructure resources to be provisioned, the configuration required, and the dependencies between resources.
2. Implement an infrastructure as code tool, such as Terraform or CloudFormation, to automate the creation and management of infrastructure resources based on the defined workflow. This will allow for the infrastructure to be defined in code, which can be version controlled and easily replicated across different environments.
3. Continuously monitor and optimize the infrastructure creation process to improve efficiency and reliability. This can include identifying and addressing bottlenecks, automating additional tasks, and implementing best practices for infrastructure management.

5. **Customer Support Responses**:
   Manually responding to customer support requests is considered toil because it is a repetitive and time-consuming task that can be prone to errors. Automating the customer support response process can improve response time, consistency, and accuracy of responses, while reducing the workload on the support team.

Steps:

1. Analyze common customer support requests to identify patterns and develop responses.
2. Develop a chatbot or other automated response system that integrates with customer support channels such as email or chat.
3. Train the automated response system using machine learning techniques to improve the accuracy and relevance of responses over time.
4. Configure the system to escalate complex or unusual requests to human support agents for resolution.
5. Monitor the system to ensure it is providing accurate and relevant responses and adjust as needed to improve its performance.

Justification for automation:

- Improves response time, consistency, and accuracy of responses.
- Reduces the workload on the support team.
- Enables faster resolution of customer issues.
- Improves overall customer experience.

6. Low Storage

Description: Currently, storage space checks are performed manually without any threshold or trigger alert to notify technicians of low storage. This can lead to potential data loss or other storage-related issues. To address this problem, an automated solution is necessary.

We propose implementing a script that performs the following actions:

1. Trigger an alert when storage space reaches a low level.
2. Free up storage space by deleting unnecessary files or moving them to external storage.
3. Alternatively, send an email alert to notify technicians of the low storage.

By automating this task, technicians will no longer need to manually check for low storage, freeing up their time to focus on other tasks. Additionally, this solution will prevent any potential data loss or other storage-related issues, which can have a high impact on the business.

7. System Resource Management

Description: In the actual architecture, systems often consist of physical servers which can make it difficult to plan and manage capacity. By implementing a capacity planning process and transitioning to virtual machines (VMs), resource management can become more flexible and easier to scale as necessary. This can lead to more efficient use of resources and cost savings.

Steps:

1. Analyze the current system architecture and identify areas where capacity planning can be improved.
2. Develop a plan for transitioning physical servers to virtual machines.
3. Implement the virtual machine infrastructure and configure the necessary resources.
4. Monitor system resource usage and adjust as necessary.
5. Conduct periodic capacity planning reviews to ensure that the system is adequately provisioned, and resources are being used efficiently.

Justification: Physical servers can be expensive to deploy and maintain and can be difficult to manage effectively. By transitioning to virtual machines, organizations can improve their resource management capabilities and achieve greater efficiency. This can lead to cost savings and improved performance and can help organizations better meet their business objectives

Automation Implementation

1. Deployment Process:

```
# Define deployment script
def deploy_app(app_name, version):
    # Clone app code from repo
    git clone https://github.com/myapp/myapp.git
    # Checkout desired version
    git checkout tags/{version}
    # Build app code
    docker build -t myapp:{version} .
    # Push app image to registry
    docker push myregistry/myapp:{version}
    # Deploy app to Kubernetes cluster
    kubectl apply -f myapp-deployment.yaml
    # Wait for deployment to complete
    kubectl rollout status deployment/myapp
    # Get app endpoint
    kubectl get svc myapp

# Call deployment script with desired app name and version
deploy_app("myapp", "v1.2.3")

```

The pseudocode is a **deploy_app** function that automates the deployment process for a given app name and version. It clones the app code from a Git repository, checks out the desired version, builds a Docker image, pushes the image to a registry, deploys the app to a Kubernetes cluster, waits for the deployment to complete, and returns the app endpoint.

2. Low Storage:

```
# Define low storage alert script
def check_storage():
    # Get current storage usage
    storage = get_storage_usage()
    # Set threshold for low storage alert
    threshold = 80
    # If storage usage is above threshold, send alert
    if storage > threshold:
        send_alert("Low storage alert", f"Storage usage is {storage}%")

# Call low storage alert script periodically
while True:
    check_storage()
    sleep(3600)

```

This pseudocode is a **check_storage** function that checks the current storage usage and sends an alert if it is above a threshold. It uses a **get_storage_usage** function to get the current storage usage and a **send_alert** function to send an alert if the storage usage is above the threshold. It then calls this function periodically using a **while** loop and a **sleep** function to wait between checks.

3. System Monitoring and Alerting:

```
# Define system monitoring script
def monitor_system():
    # Get system metrics
    cpu_usage = get_cpu_usage()
    memory_usage = get_memory_usage()
    disk_usage = get_disk_usage()
    # Set thresholds for alerting on high usage
    cpu_threshold = 90
    memory_threshold = 80
    disk_threshold = 80
    # If any metrics are above their threshold, send alert
    if cpu_usage > cpu_threshold:
        send_alert("High CPU usage alert", f"CPU usage is {cpu_usage}%")
    if memory_usage > memory_threshold:
        send_alert("High memory usage alert", f"Memory usage is {memory_usage}%")
    if disk_usage > disk_threshold:
        send_alert("High disk usage alert", f"Disk usage is {disk_usage}%")

# Call system monitoring script periodically
while True:
    monitor_system()
    sleep(60)

```

The pseudocode describes a **monitor_system** function that gets the current system metrics (CPU usage, memory usage, and disk usage) and sends an alert if any of them are above their threshold. It uses **get_cpu_usage**, **get_memory_usage**, and **get_disk_usage** functions to get the current system metrics and a **send_alert** function to send an alert if any of the metrics are above their threshold. It then calls this function periodically using a **while** loop and a **sleep** function to wait between checks.