# Project:
# Plan, Reduce, Repeat

## Directions and Submission Template

*[Vyacheslav Khvan]*:
*[11/24/2022]*

# Overview:

**You have recently joined the SRE team for an exotic plant 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
# Release 1

## Stakeholders

- Developers
  - John Doe
  - Jane 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
  - Vyacheslav Khvan

## Code Changes

- Feature additions
  - Rearranged the catalog menu in the UI to accommodate the additional catalog, as well as make it more user-friendly (Tk-202)
- Security fixes
  - Fixed a security flaw where attackers could execute a SQL injection attack (Tk-205)

## Data and System Changes

- Data changes
  - Added new tables in the database to handle the additional catalogs (Tk-203)
- System changes
  - Added an additional component to the application, an order processor (Tk-201)

# As-Built Doc
# Release 2

## Design decision highlights

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.

## Test Section

- 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 8 Gb of RAM.
    - The component runs with 4 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.

# As-Built Doc
# Release 2

- 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.

## Deployment Notes

Download the latest code, build image, shut down the app, run the database migrations, apply Kubernetes deployment manifest.

# Deployment File Release 2

Update the file for Release 2 to match the description in the scenario:

```
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: 320Mi
            cpu: 250m
        ports:
        - containerPort: 80
      - name: order_processor
        image: nginx:latest
        resources:
          requests:
            memory: 8Gi
            cpu: 4
        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.

### SOP

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.

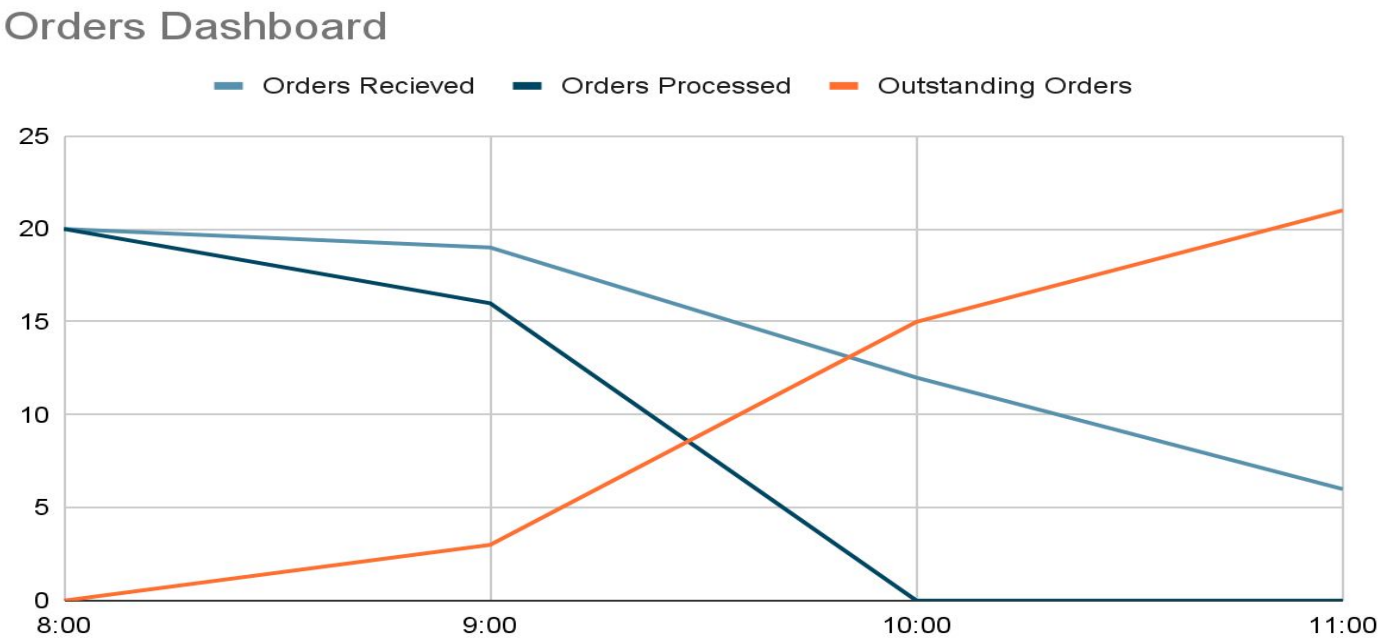
There are logs at /home/sre/course4/order_processing.log. If the server is not overloaded, this is a good place to check for errors. If you encounter any errors, send a message to the developers so that they can troubleshoot.

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

### Summary

You receive an alert that the storage is running out on the mount where application logs are being written to. After consulting the SOP, you reach out to the team responsible for the server. They respond that Steve is normally in charge of handling the logs. Every morning he would run the commands listed in the run book, but he has been out sick for a week. The other members of the team forgot that it needed to be done, so the mount filled up.

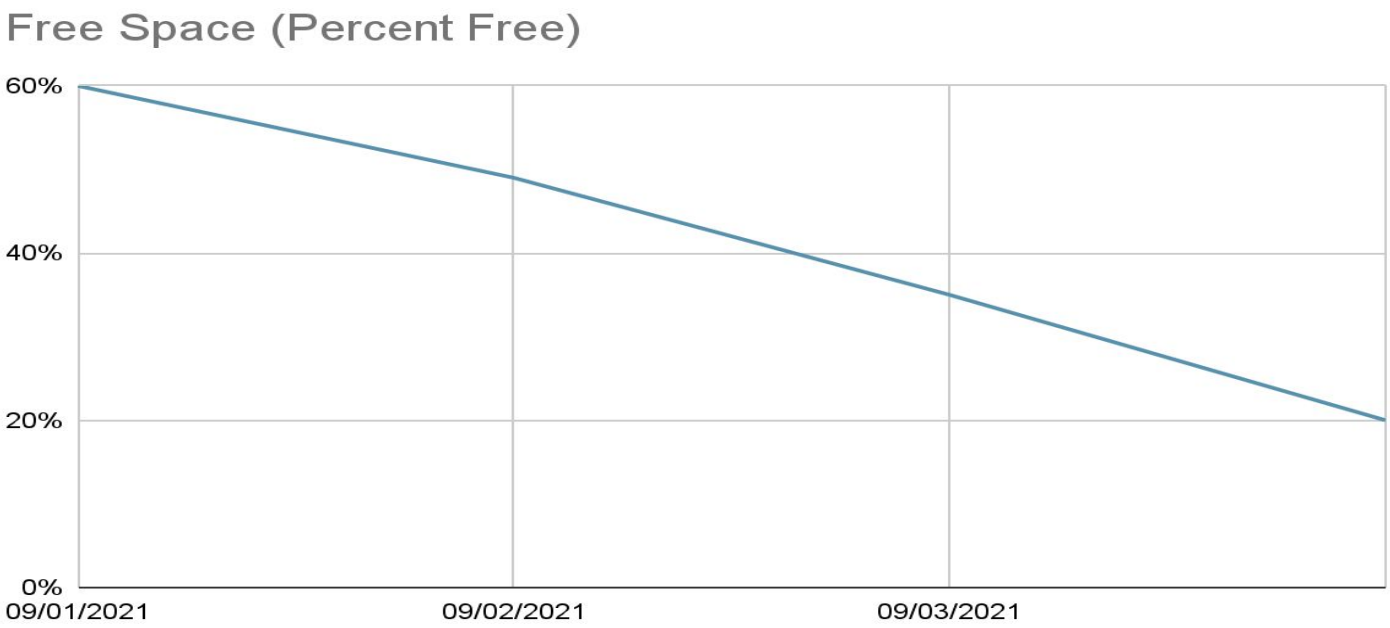
### SOP

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.

### Log/Monitoring Details



# On-Call Shift -- Alert 3

## DNS Troubles

### Summary

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.

### SOP

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.

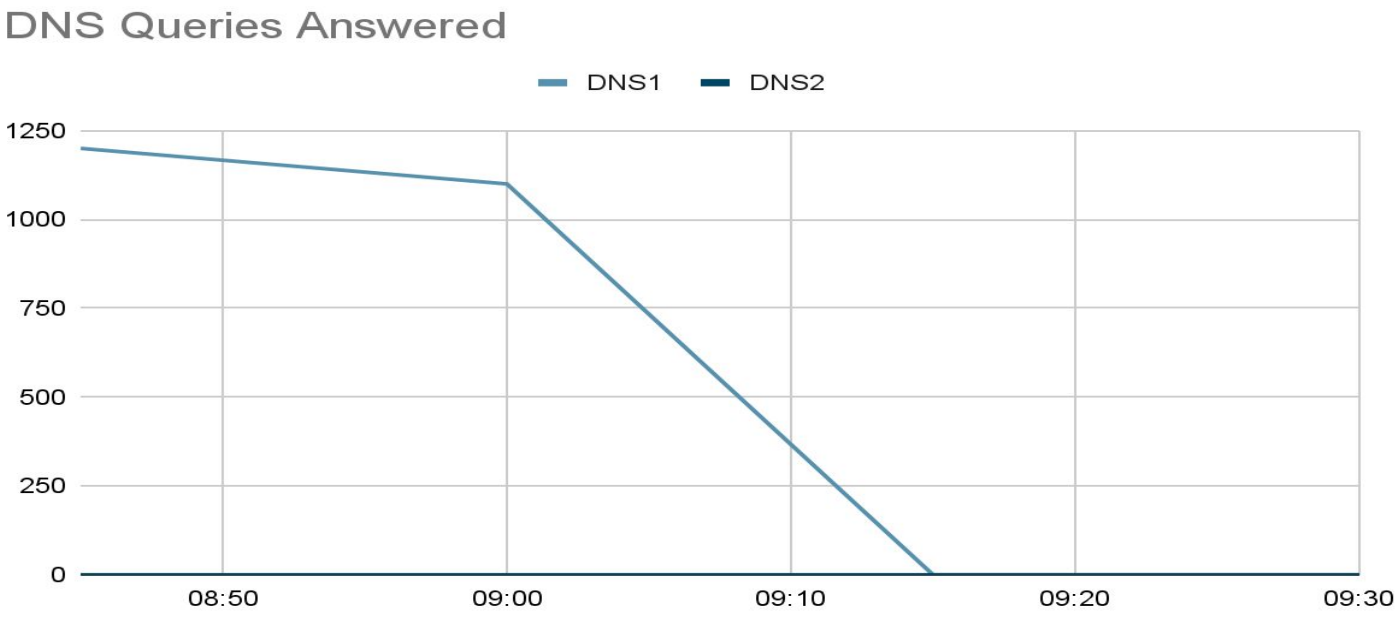
Failover Procedure

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

### Log/Monitoring Details

```
Answering Query.
Answering Query.
Answering Query.
Answering Query.
Answering Query.
Answering Query.
Answering Query.
Answering Query.
Answering Query.
Answering Query.
Answering Query.
Answering Query.
Answering Query.
Answering Query.
Unexpected Error encountered.
Unexpected Error encountered.
Unexpected Error encountered.
Unexpected Error encountered.
Unexpected Error encountered.
Unexpected Error encountered.
Unexpected Error encountered.
Unexpected Error encountered.
Unexpected Error encountered.
```

# On-Call Shift -- Alert 4

## Application Outage

## Summary

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

## SOP

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

## Log/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

### Log/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
```

# On-Call Shift -- Alert 4

## Application Outage, cont

### Log/Monitoring, cont.

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
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

## 11.28.2022/10:15 PM -- *Order Processing Issues*

### Troubleshooting

- After receiving the alert, the Orders Dashboard was checked.
- At 9:00 a.m. there was an increase in Outstanding Orders, while the number of Received and Processed Orders began to decrease.
- The Order Processor was not overloaded.
- The logs /home/sre/course4/order_processing.log were checked and errors were found processing orders (Error Processing Order. Error #12).

### Resolution

- Restarted server with Order Processor Application.
- The Order Processor continued where it left off.
- A bug report has been sent to the developers.
- It is necessary to configure monitoring to detect such errors (Future).
- Use retries logic for batch processing (Future).

## 11.28.2022/12:37 PM -- *Low Storage Alert*

### Troubleshooting

- After receiving the alert, the Storage Dashboard was checked.
- Free space was below 20%.
- Not enough storage for log /home/sre/course4/app_log.

### Resolution

- Contacted compliance, they said what logs can be cleared.
- Logs cleared.
- It is necessary to automate the process of cleaning logs (Future).

# On-Call Summary Log

## 11.28.2022/14:10 PM -- *DNS Troubles*

### Troubleshooting

- After receiving the alert, the DNS Queries Answered Dashboard was checked.
- DNS1 is the current server answering requests.
- Checked DNS1 is reachable - DNS1 and DNS2 are reachable.
- Checked the logs - found errors in the logs (Unexpected Error encountered).

### Resolution

- Determine the active server with the dnsTool.
- Successfully shutdown DNS1 server.
- Started the failover to DNS2 server.
- Engaged the Networking team to bring the standby server back online.

## 11.29.2022/09:00 PM -- *Application Outage*

### Troubleshooting

- Verified the application is indeed unreachable.
- Checked the hosts are up and the application processes are running.
- Started escalation procedure.
- Contacted with Customer Support -- Susan Vega, Networking -- Bob Sparrow, Contacted with Ops -- Glen Hammer, Contacted with Database Admin -- Karen House.
- Bob Sparrow checked the networking, it Ok.
- Karen House checked the database, it Ok.
- Glen Hammer checked the infrastructure, it Ok.
- Glen Hammer recommended restart application.
- The application has been restarted.
- Application is still not respond.
- Glen Hammer and Gal Tree identified that QA build of application was deployed to production by mistake.

### Resolution

- Redeploy production build to production environment.
- The application is reachable and respond.

# Post-Mortem

## *Application Outage -- 11.29.2022/09:00 PM*

### Stakeholders

Customer Support -- Susan Vega
Networking -- Bob Sparrow
Ops -- Glen Hammer
Database Admin -- Karen House
Development Team -- Gal Tree
SRE -- Vyacheslav Khvan

### Incident Timeline

09.15 Susan Vega reported of an application outage
09:30 Bob Sparrow checked networking - no errors found
09:32 Karen House checked database - no errors found
09:35 Glen Hammer checked infrastructure - no errors found
09:37 Gal Tree checked logs
09:45 The application has been down
09:47 The application has been running
09:55 Susan Vega reported of application is still not respond
10:07 Glen Hammer and Gal Tree identified that QA version of application was deployed to production by mistake
10:15 The application has been down
10:17 Redeploy production build to production environment
10:23 The application has been running
10:30 The application is responding

### Impact

The application 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.

# Post-Mortem

## Resolution

Glen Hammer and Gal Tree identified that QA version of application was deployed to production by mistake. QA version of application was taken down of production and production build was deployed to production environment.

## Action Plan

It is necessary to use GitOps approaches and add code review for Helm Chart, thus, all configuration changes will be versioned. Also need to add a staging environment.

## 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. The established release process is a manual affair generally done by one of the operations team members. This is a manual process that makes deploying new releases more difficult. Increases the number of possible errors in the deployment process. Generally contrary to DevOps values.
2. The absence of a configuration management system leads to many problems. 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.
3. 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.
4. Manual rotation of logs leads to system incidents of insufficient storage space. This is extra work that takes up the time of one of the engineers every morning.
5. One of the DNS servers often goes down. The failover procedure is performed manually. With frequent incidents, this will turn into toil.

# Toil Reduction Plan

## Strategies for Eliminating Toil

1. Set up CI/CD for release process with GitOps approaches. This automates the deployment process and reduces errors. It will also speed up feedback.
2. Use a configuration management system and tools, such Ansible and Helm to manage Kubernetes resources and infrastructure. With the use of a version control system, configuration management will become more transparent and consistent.
3. Implement best practices in capacity management. Use Tiered Capacity Management, create Emergency capacity tier, Analyze system requirements before release. Use Horizontal Scaling for Kubernetes Deployment.
4. Automate log rotate with logrotate, Filebeat and ELK stack. Store logs to ElasticSearch and set up Observability to Kibana.
5. Automate DNS failover procedure with Apache Airflow.

# Automation Implementation

*Continuous Integration with GitHub Actions and DockerHub*

```
name: Main Application
on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - name: Checkout
      uses: actions/checkout@v2
    - name: Set up Python 3.8
      uses: actions/setup-python@v2
      with:
        python-version: 3.8
    - name: Set up QEMU
      uses: docker/setup-qemu-action@v1
    - name: Set up Docker Buildx
      uses: docker/setup-buildx-action@v1
    - name: Login to DockerHub
      uses: docker/login-action@v1
      with:
        username: ${{ secrets.DOCKERHUB_USERNAME  }}
        password: ${{ secrets.DOCKERHUB_TOKEN }}
    - name: Login to GHCR
      if: github.event_name != 'pull_request'
      uses: docker/login-action@v1
      with:
        registry: ghcr.io
        username: ${{ github.repository_owner }}
        password: ${{ secrets.GHCR_TOKEN }}
    - name: Build and Push
      uses: docker/build-push-action@v2
      with:
        file: ./Dockerfile
        platforms: linux/amd64
        push: ${{ github.event_name != 'pull_request' }}
```

# Automation Implementation

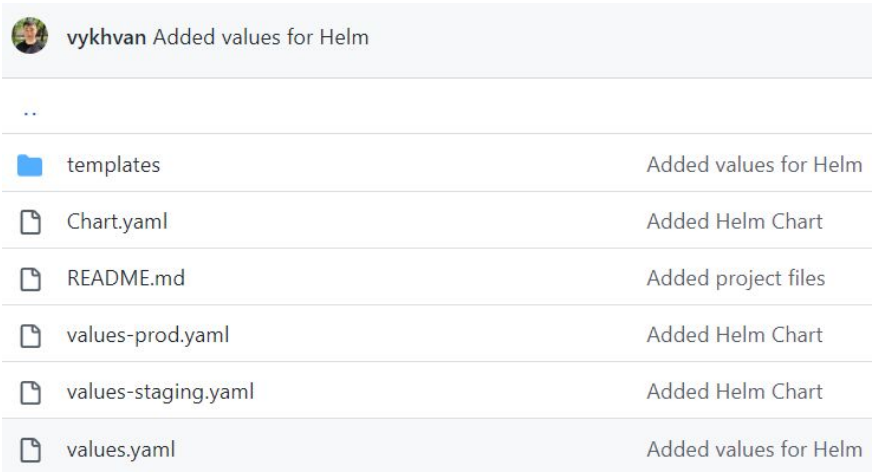
*Continuous Deployment with ArgoCD Application*

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
 name: main-application-staging
 namespace: argocd
spec:
 destination:
   namespace: default
   server: https://kubernetes.default.svc
 project: default
 source:
   helm:
     valueFiles:
     - values-staging.yaml
   path: helm
   repoURL: https://github.com/exotic-plan/main-application
   targetRevision: HEAD
```

# Automation Implementation

*Helm Chart to manage Kubernetes resources and infrastructure*

```
ApiVersion: apps/v1
kind: Deployment
metadata:
  name: app-deployment
  namespace: course4
  labels:
    app: mainApp
spec:
  Replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app: mainApp
  template:
    metadata:
      labels:
        app: mainApp
    spec:
      containers:
      - name: mainApp
        image: {{ .Values.mainApp.image.repo }}:{{ .Values.mainApp.image.tag }}
        resources:
          requests:
            memory: {{ .Values.mainApp.resources.requests.memory }}
            cpu: {{ .Values.mainApp.resources.requests.cpu }}
        ports:
        - containerPort: {{ .Values.mainApp.containerPort }}
      - name: order_processor
        image: {{ .Values.orderProcess.image.repo }}:{{ .Values.orderProcess.image.tag }}
        resources:
          requests:
            memory: {{ .Values.orderProcess.resources.requests.memory }}
            cpu: {{ .Values.orderProcess.resources.requests.cpu }}
        ports:
        - containerPort: {{ .Values.orderProcess.containerPort }}
```



# Automation Implementation

*Apache Airflow DAG for DNS Failover Procedure*

```
from datetime import datetime, timedelta
from airflow import DAG
from airflow.providers.ssh.operators.ssh import SSHOperator
from airflow.providers.http.sensors.http import HttpSensor

default_args = {
    "owner": "Vyacheslav.Khvan",
    "depends_on_past": False,
    "retries": 3,
    "retry_delay": timedelta(minutes=10),
}

with DAG(
    "DNS Failover Procedure",
    start_date=datetime(2022, 12, 12),
    max_active_runs=3,
    default_args=default_args,
    catchup=False,
) as dag:

    tasks = {}

    def check(response):
        if response == 200:
            print("Returning True")
            return True
        else:
            print("Returning False")
            return False

    tasks["poking"] = HttpSensor(
        task_id='poking',
        http_conn_id='dns-host',
        method='GET',
        endpoint='',
        data={ "command": "run" },
        response_check=lambda response: True if check(response.status_code) is True else False,
    )

    tasks["failover"] = SSHOperator(
        task_id="failover",
        ssh_conn_id="dns-host",
        command="""if shutdown then dnsTool -a shutdown -s dns1;
                   else dnsTool -a failover -s dns2""",
    )

    tasks["poking"] >> tasks["failover"]
```

# Automation Implementation

*Automate log rotation with lograte and Filebeat*

*logrotate.conf*

```
/var/log/my-server/my-server.log {
    daily
    missingok
    rotate 7
    notifempty
}
```

*filebeat.yaml*

```
filebeat.inputs:
- type: filestream
  id: main-application
  paths:
  - /home/sre/course4/app_log*
```