IP EIS

Automation of Oracle Databases

RedHat Ansible POC Demo – Bulk Patching a Fleet of Oracle 21c Servers with Opatch

May 2024

Presented by Vladimir.Grigorian@jpmchase.com

JPMorgan Chase & Co.

V20240803_005

Contents

vladimir.grigorian@jpmchase.com.

What is Automation?

Automation is a human practice of delegating all processes that are likely to be repeated to robots

Don't We Use Automation Already?

If we used automation, we would know. Automation would have been everywhere, so are the metrics and standards enforcement because that is what it takes to gid rid of inefficiency. Instead, the "automation" we use is usually requested by the end technology users for their own convenience. We are shortcutting and shell-scripting subtasks, instead of automating entire business processes to free up the time to concentrate on more crucial business aspects.

The productive way is to use it as a part of the Infrastructure as Code implementation Triad (IaC), which is always enforced by the management downwards as a planned & controlled endeavor, and done for enterprise efficiency, not just mere technology user convenience.

What is the Infrastructure as Code Triad Implementation?

1. Standardization
2. Automation
3. Metrics

(…they only work together!)

Why do have to automate at all?

Today's IT Challenges

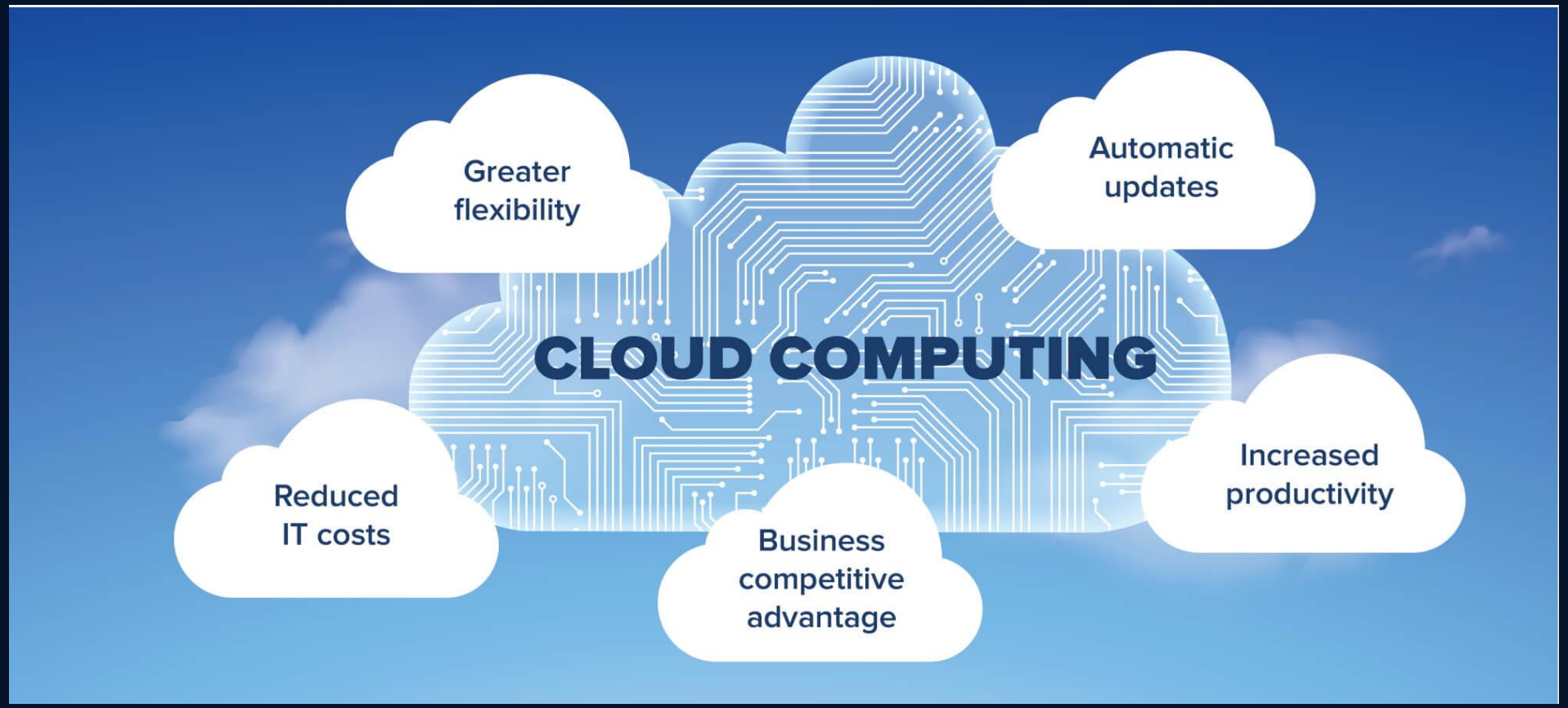
Line of Business
Challenged to deliver services faster, at scale and more efficiently

Developers
Need to develop applications faster with greater productivity (number of new applications per year increased by 35% since 2020

IT Operations
Must provide infrastructure on-demand that scales as needed (i.e., provisioning improved by 40%, change management 44%, and patching by 41%)

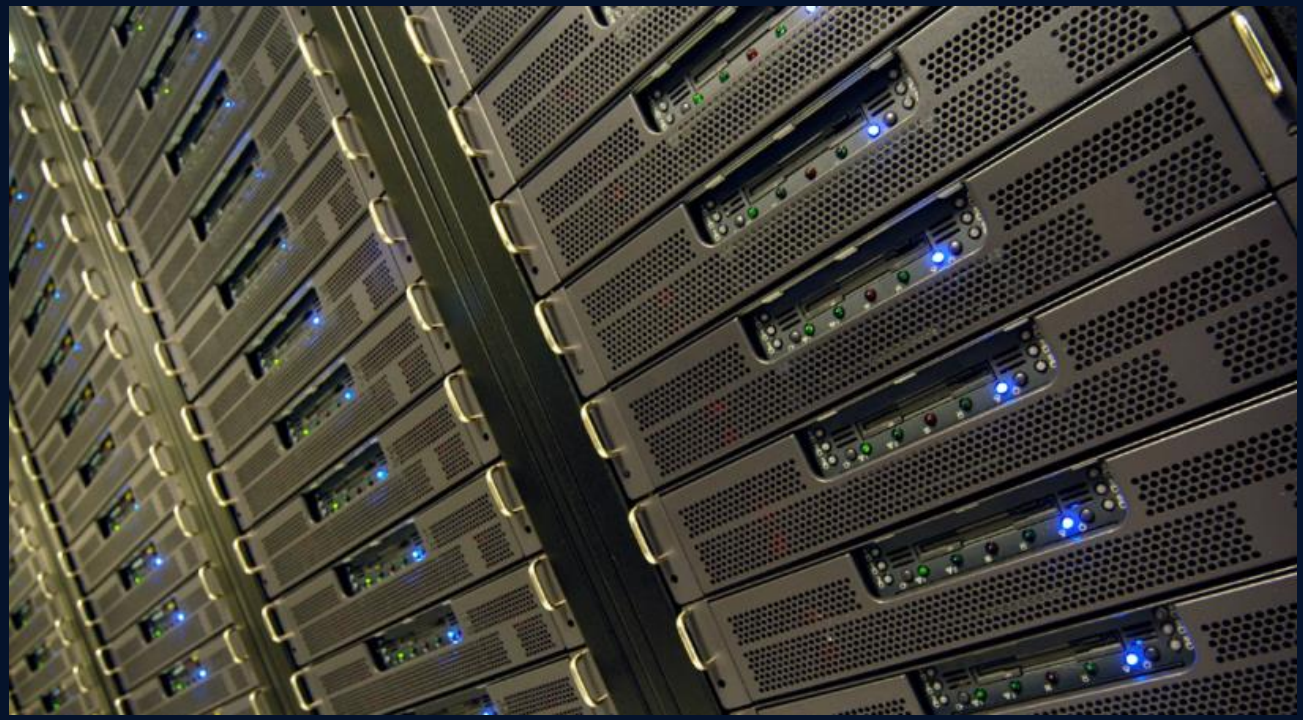
Why do have to automate at all?

It Is All Going to Be in the Cloud Soon…



Why do have to automate at all?

So, most of these (Actual JPMC Servers from our Langan DC, NJ)



Why do have to automate at all?

…will be replaced by this (JSON infrastructure definition code)

IP EIS

Benefits of Automation :

Low Cost

You don't have to hire a team of professionals to routinely manually manage resource provisioning, configuration, troubleshooting, hardware setup, and so on. Automation saves time and money.

Accountability

When you need to trace changes to definition files, you can do it with ease. They are versioned, and therefore all changes are recorded for your review at a later point. So, once again, there's never any confusion on who did what.

High Speed

IaC automates resource provisioning across environments, from development to deployment, by simply running a script. It drastically accelerates the IT infra life-cycle and makes your organization more responsive to external challenges.

Resilience

Resource provisioning and configuration is often a labor-intensive and skill-intensive task, usually handled by skilled resources. When one or more of them leave the organization, they take their knowledge with them. With IaC, resource provisioning intelligence remains with the organization in the form of definition files.

High Quality

People commit mistakes. That's a fact. No matter much effort your team puts in, they are bound to make errors – several of them, in fact. In the case of IaC, the definition files are a single source of truth. There's never any confusion about what they do. You execute them repeatedly and get predictable results every time.

Scalability

The load current DBA team carries can be increased by 5%-10%. If you want more, you have to hire more DBAs. With IaC implementation, you may increase your workload in a matter of magnitudes and even exponentially in limited cases, before you even start considering a CTRL node upgrade.

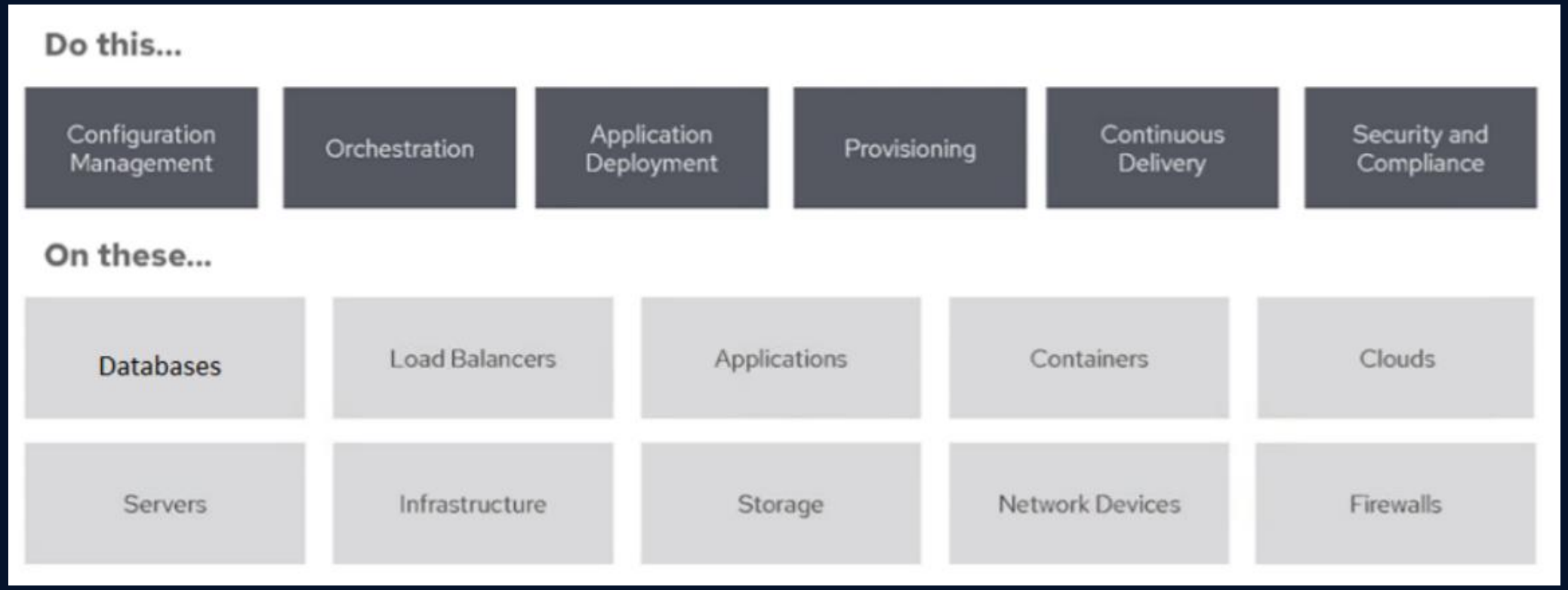
What is usually automated?

Not every DBA task can or should be automated. Depending on the type of a shop, around 30% of regular activities are candidates. Anything involving pre-set tasks that ends with a known deliverable - migrations, deployments, upgrades, patching, load testing, backup/recovery tests, DR switchover tests, etc. Automation may involve basic decision making and rollback capabilities, as we will see from the patching demo.

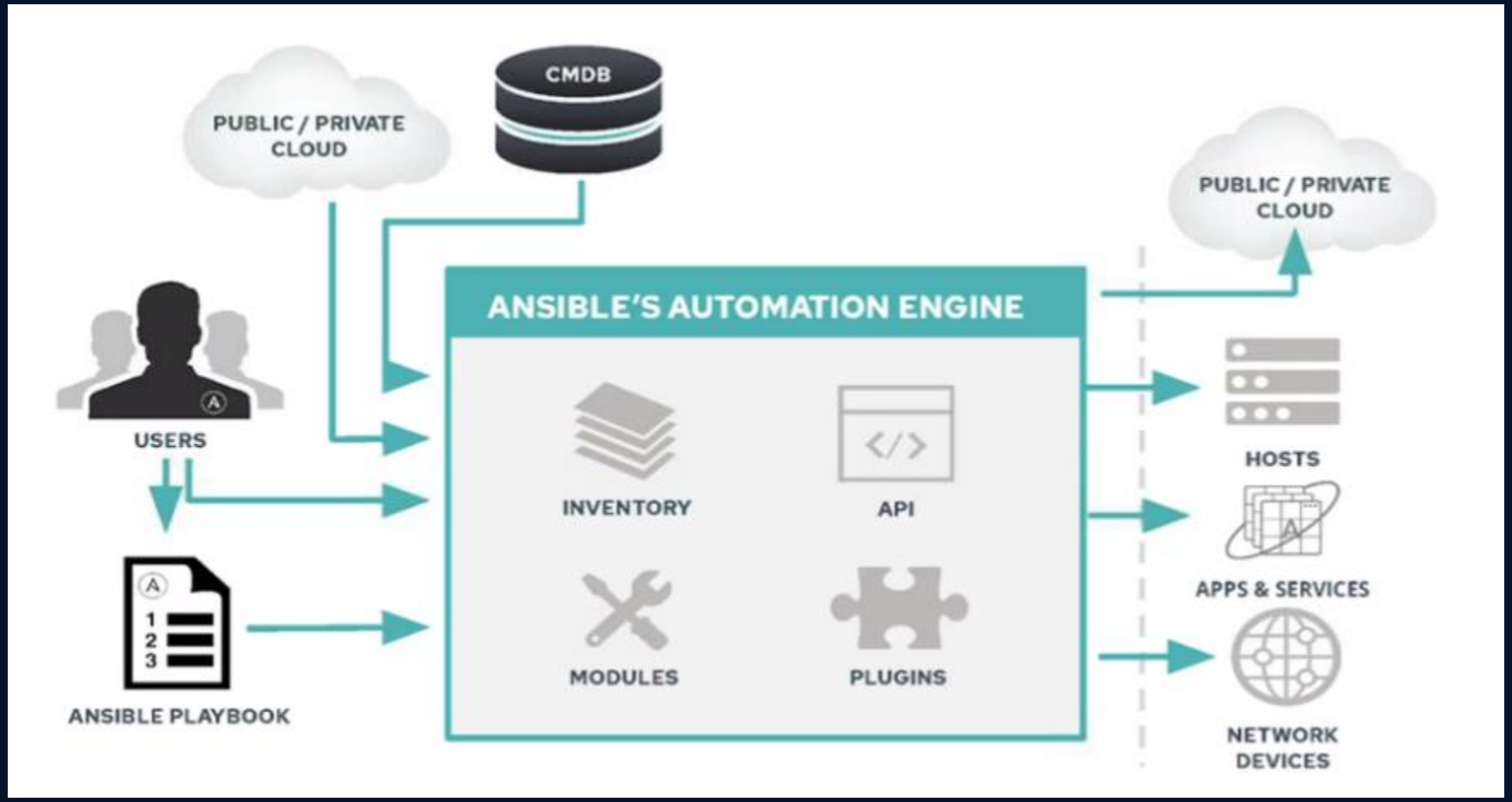
What isn't Automation Material?

Anything involving intuition (performance issues, exploratory troubleshooting, forecasting, capacity planning, etc.) isn't a good automation material.

What Can Ansible Automation do for us?



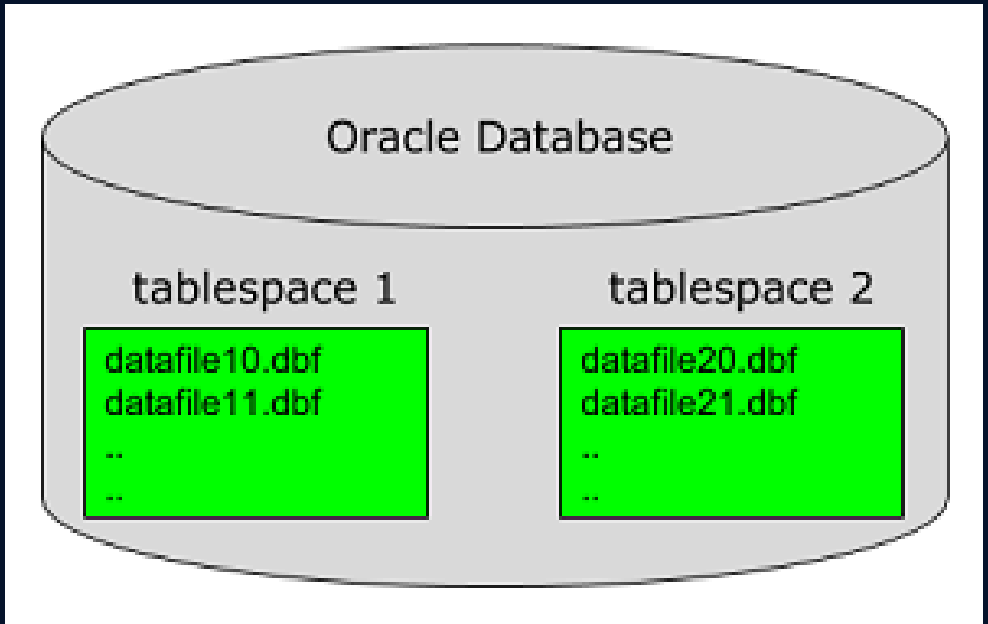
What Can Ansible Automation do for us?



An Example of Automation - Tablespace out of Space

Automation allows us to keep doing what we have been doing, but faster, better and much more scalable, even the tasks we consider our DBA bread and butter, the tasks we have been doing the same way for decades.

A case in point: Datafiles Addition to Tablespaces in an Oracle database. This is the most common reason an on-call DBA is paged in the middle of the night for. Lets compare how a usual Oracle database administration approach compares to an automated one.

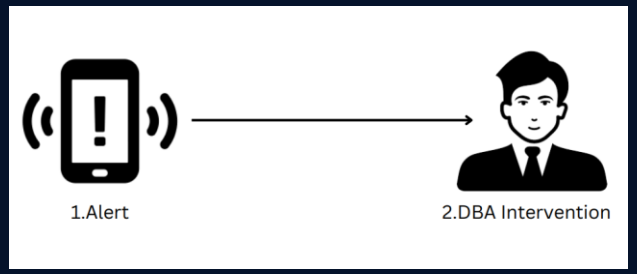


An Example of Automation - Tablespace out of Space

An approach to automating this process – how often do we currently do this? How well do we do it? If a situation calls, how many of these can we handle at the same time and for how long? How scalable?

How often: 40man/hours/week per team. A good 10% of our regular tasks are for a TBS "out of space" alert, and over 30% of the DBA on-call pages. For a team of 10 DBAs with an on-call rotation means at least 40-50 man/hours a week spent on repeating an identical task in different systems or context for years.

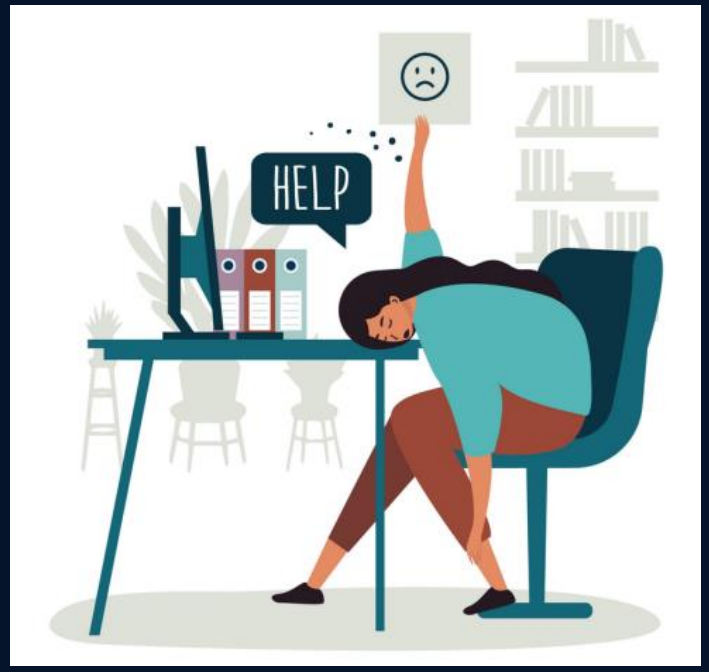
How well: (ave quality score): 85%. We always do the same minimal thing – get the page and add a datafile. All of these defects may diminish the work quality score: How long was the app down without the TBS able to grow? Forgot a step? Added too much? Then other databases starve for space, while this DB will never use as much. Didn't add enough space to the TBS? Will get paged again in two hours. Didn't follow the naming convention? Added the wrong type of storage, to the wrong DB, etc. All of these errors are quality hits, aberrations.



An Example of Automation - Tablespace out of Space

How scalable is the process: the DBA currently can handle a normal load without issues, provided the on-call rotation isn't more often than twice a month and the load doesn't exceed the 4-5hrs/week on-call. Can handle 20% increase in duties for a short period only (a few days). Try to double the load and our happy DBA will turn into a helpless mess in a week:

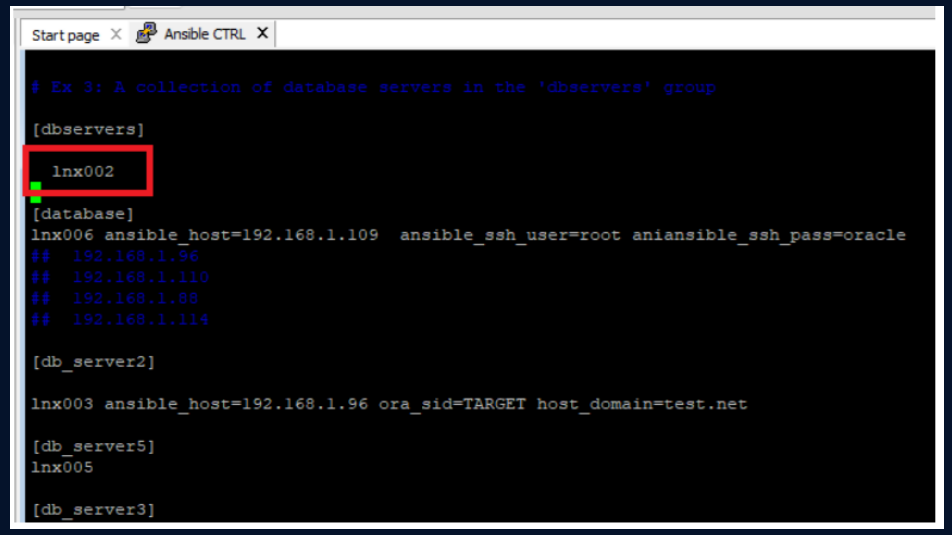
Regular workload



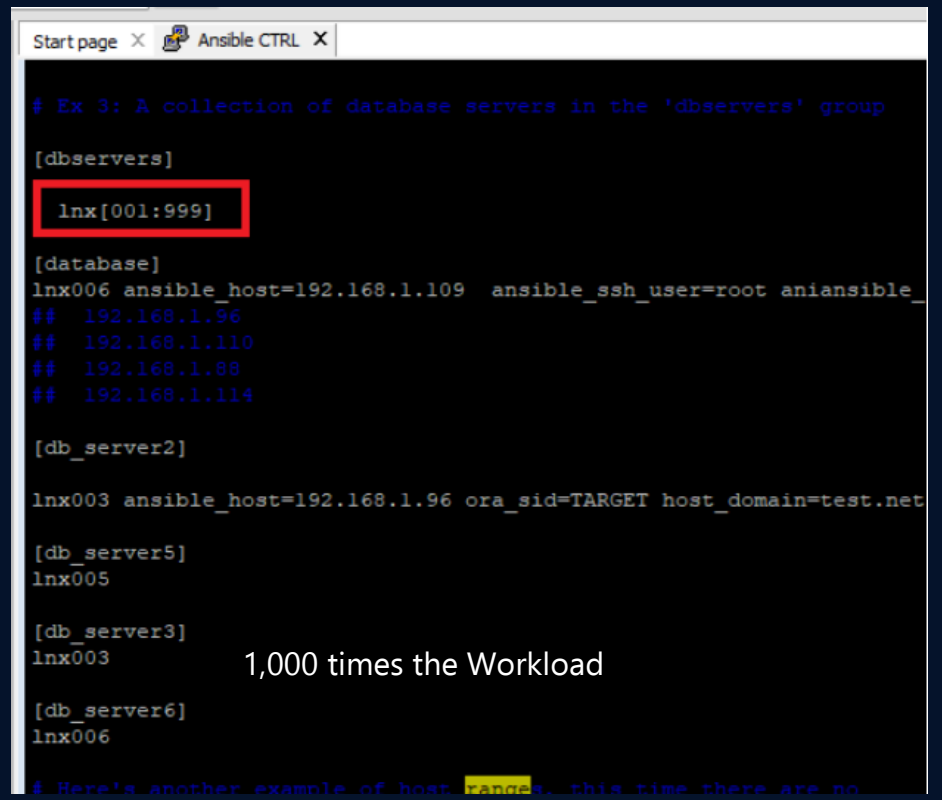
Doubled workload

An Example of Automation - Tablespace out of Space

Now, once we captured the current metrics for the TBS process (the speed: 35mins, the quality: 85%, the amount of time spent: 15 mins, the scalability factor: 120% burst for no more than two days in a row), lets see how well will automation do with the same metrics.
Lets start with scalability. Below is a screenshot of an Ansible HOSTS inventory. It is running an Ansible playbook on one server called lnx002 on the left, and a thousand of them (from lnx001 all the way to lnx999 and all the servers in between). That is the only change it requires to start doing a job of a 1,000 DBAs as opposed to just one.



Regular workload



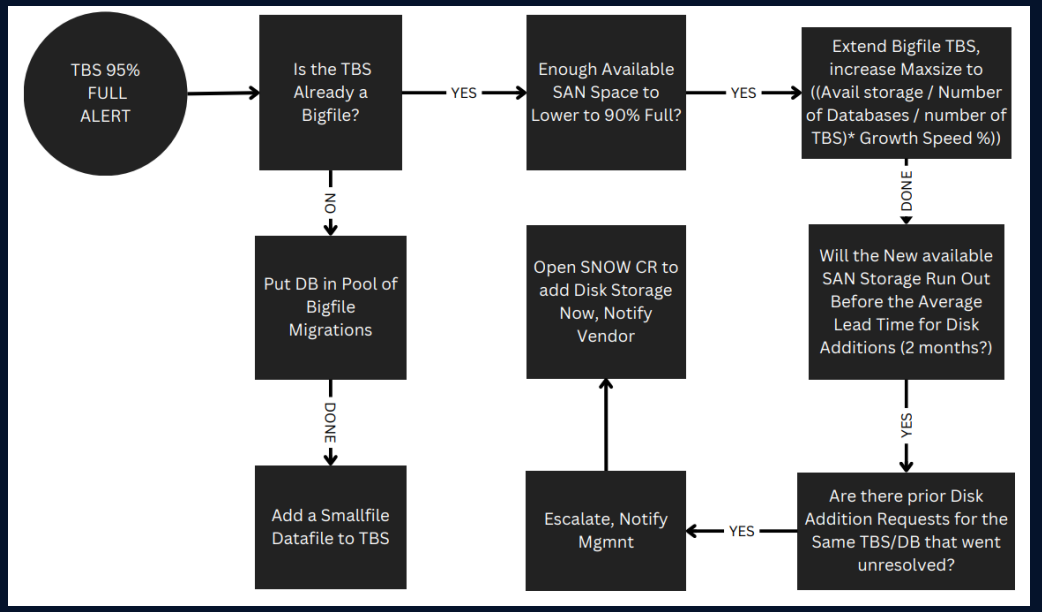
1,000 times the Workload

An Example of Automation - Tablespace out of Space

Now, lets compare the quality. Unlike a normal DBA issue resolution, the automated solution takes advantage of the SNOW/Ansible API. The DBA doesn't get pages in the middle of the night for TBs out-of-space alerts anymore. Ansible resolves the error automatically, according to Oracle's best practices, in seconds. It also checks all the SAN or ASM space, makes the capacity planning decisions and even orders new disks and checks on the status of previous requests. The only notifications we get is for the DBA manager, who may get an occasional alert only if a disk provisioning was not done by another team).

Unautomated TBS out of space Resolution: Ave Process Quality Score 85%

Automated TBS out of space Resolution : Ave Process Quality Score 99.92%



Summary : Which is Better?

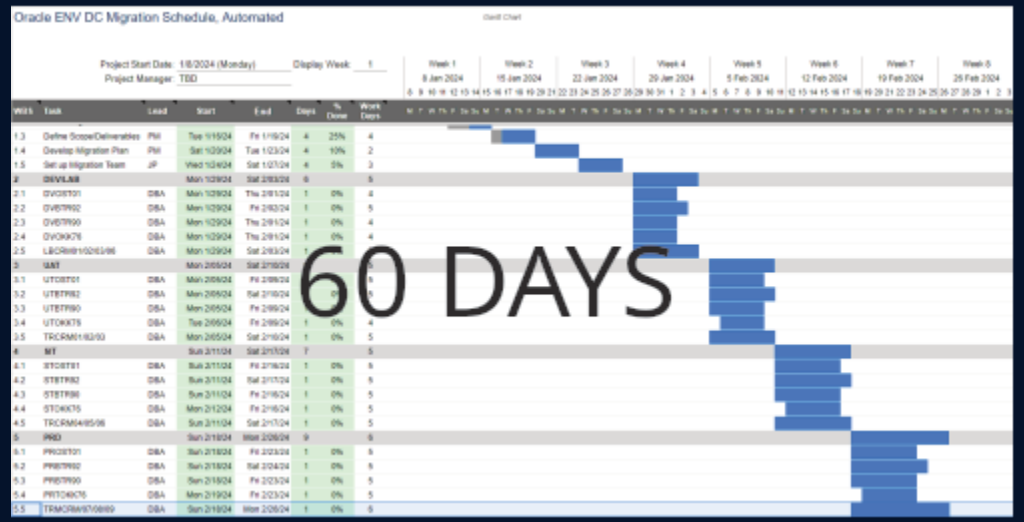
Speed: On average, it takes a DBA 35 minutes to resolve a TBS out of space situation. The automated solution, not only it diminishes the frequency of such failures in the long run, it also makes such occurrences unnoticeably quick. Should it ever occur it will be resolved within 2 minutes (one minute for status polling, one more minute to add a datafile and/or place and order for additional storage).

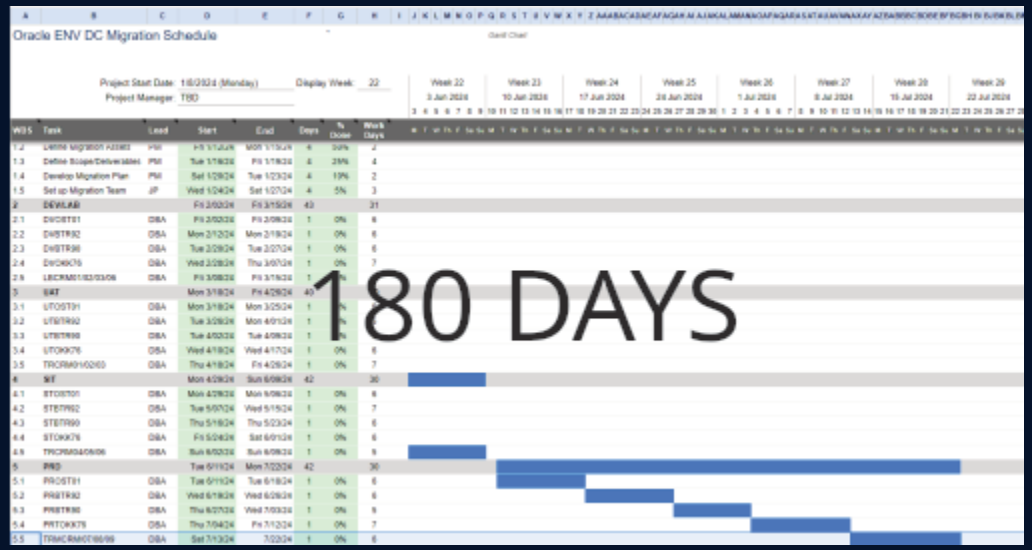
Oracle TBS Out of Space Resolution Score Comparison

Metric	On-call DBA (reactive)	Ansible (proactive)
Quality (%)	85	99.92
Ack Time (minutes)	10	0
Turnaround (minutes)	35	2
Scalability (times the workload)	1.2	1,000 + (limited by hardware)
Ave Cost (in $, per 1,000 Servers per Year)	$19,200,000	$0

Automating a Budgeted Project – How Fast a DC Migration Will Run?

Speed: Lets see how automation may affect the speed of a budgeted project, instead of a single process. In this case it is a DC migration. We will compare a course of an automated and a conventional migrations. It is obvious which is which.





How Does Automation Speeds Up Budget Projects?

This example of a cluster build task from an actual Ansible shop explains WHY some of the project work completes faster if automated. All steps marked “ANS” used to be executed by the DBAs, manually.

```
                                Cluster Build

1.[ANS] Oracle software installed   -- 1hr, automated
2.[ANS] GRID software installed     -- 1hr, automated
3.[OMS] Monitoring agent software installed. Cluster registered with OEM  --DBA, manual
4.[USG] Snap Center agents installed  --USG, external
5.[SSG] .profile copied from /data/home/oracle/host_config@<> to /data/home/oracle on each cluster --1hr, automated
6.[ANS] glogin.sql copied from /data/home/oracle/host_config@<> to $ORACLE_HOME/sqlplus/admin on each cluster node --1hr, automated
7.[ANS] ldap_oud_prod.ora copied from /data/home/oracle/host_config@<> to /var/opt/oracle on each cluster node --1hr, automated
8.[ANS] soft link created for wallet  on each cluster node  ln -s /var/opt/oracle/ldap_oud_prod.ora /var/opt/oracle/ldap.ora --1hr, automated
9.[ANS] sid.rac copied from /data/home/oracle/host_config@<> to /data/home/ooracle/local on each cluster node --1hr, automated
10.[ANS]        sqlnet.ora copied from /data/home/oracle/host_config@<> to /data/home/ooracle/local on each cluster node --1hr, automated
11.[ANS]        Create empty tnsnames.ora file in /var/opt/oracle on each node --1hr, automated
12.[MANUAL]     Create soft link from  /var/opt/oracle/tnsname.ora to $ORACLE_HOME/network/admin/tnsnames.ora   --DBA, manual
        ln -s /var/opt/oracle/tnsname.ora $ORACLE_HOME/network/admin/tnsnames.ora
        35 00 * * * /store/oracle/rsync/rsync_cronV2.sh ALTER >/dev/null 2>&1
13.[MANUAL]    Add following line to crontab   --DBA, manual
        ##CRS Log cleanup
        #
        00 3 * * * /drawers/oracle/cron/cleanup_RAC_trace.sh <hostname> >/dev/null 2>&1
14.[ANS] Oracle Standby created --2hrs, automated
15.[ANS] Empty Oracle GoldenGate installed & configured --5hrs, automated
```

The Opatch Ansible Demo

1. What: Patching Oracle 21c with an RU patch, custom code
2. Where: 2 Oracle LINUX database hosts
3. How: Ansible CTRL node, LINUX. Runs a playbook
4. Difference: infinitely more scalable, risk-free, faster patching of Oracle

vladimir.grigorian@jpmchase.com.

Why Bother Automating Oracle Patching?

If you are an Oracle DBA / Manager you know how important it is to test and apply patches. The following shows a projected metrics comparison score card for automating the process. It only assumes you apply the quarterly patches to a fleet of 1,000 database servers. Note the quality score and the yearly cost comparison.

Patching w Oracle Opatch Manually vs an Automated Solution

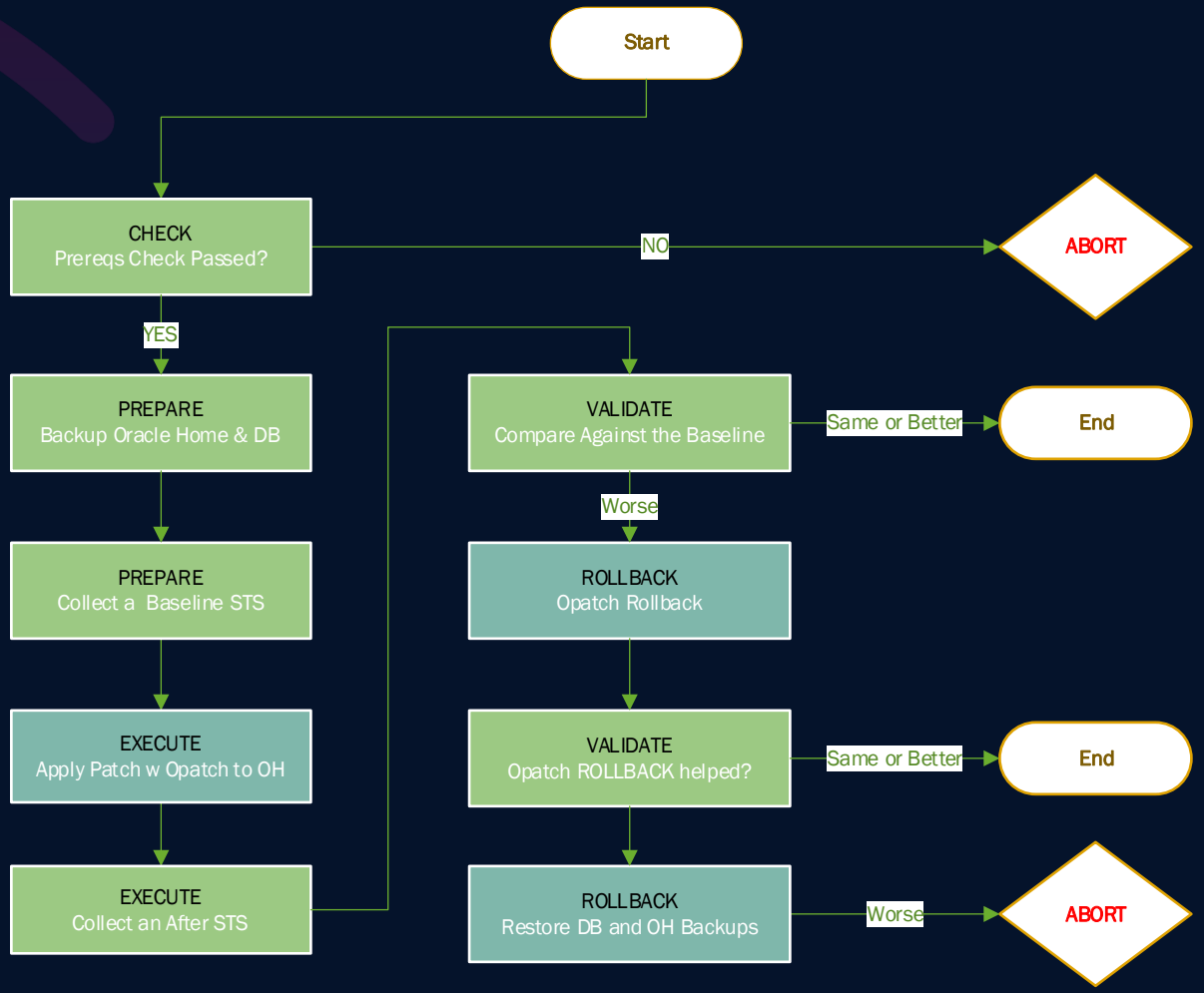
Metric	DBA (reactive)	Ansible (proactive)
Quality (%)	74	98.01
Ack Time (minutes)	NA	NA
Turnaround (days)	45	1
Scalability (times the workload)	1.2	1,000 + (limited by hardware)
Ave Cost (in $, per 1,000 Servers per Year)	$115,200,000	$10,000 (code maintenance)

Explanation of the Demo

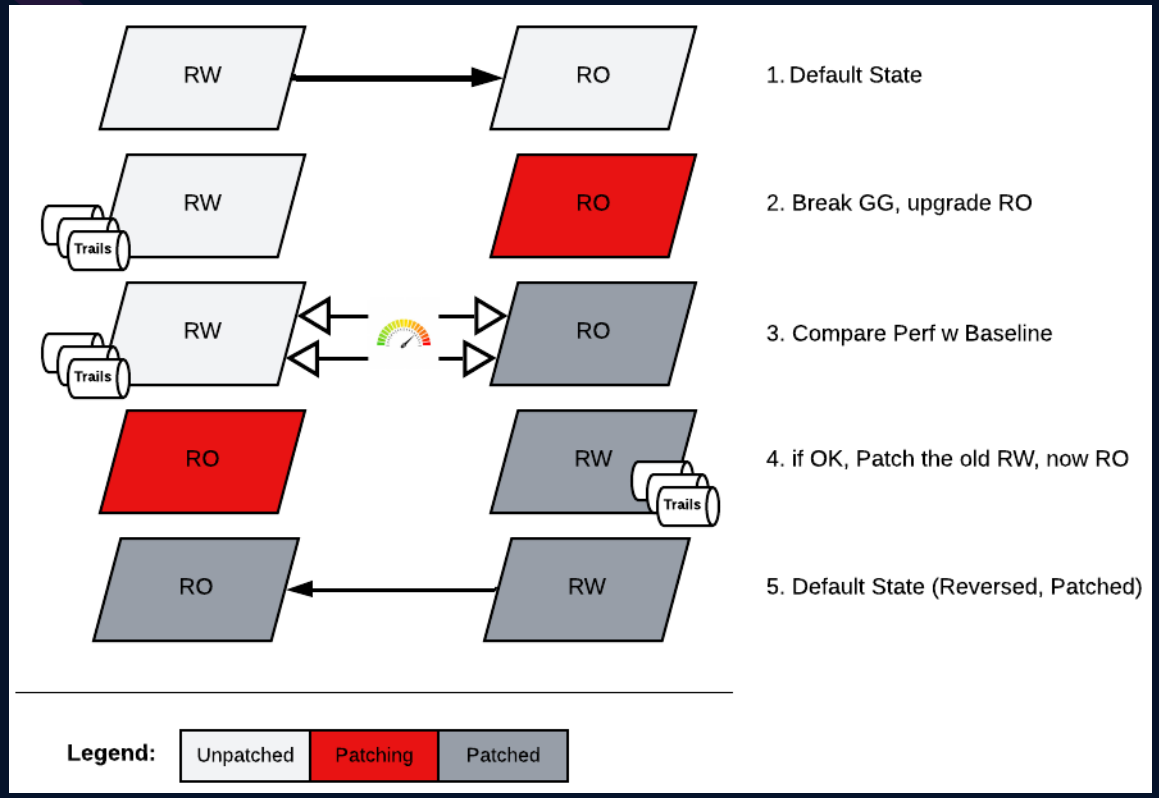
Here is a flowchart for such an automated Ansible patching. For a fleet of 100 servers this is a 6 months endeavor. Why that long? Multiple environments, the DBAs have other work to attend to, they don't work weekends either.

The automation boils the fleet patching down to just 14 hours. It can be started by pressing the "Approved" button for a patching request in ServiceNow (Ansible has a SNOW ticketing API). The Oracle patching then kicked off on one server or one thousand servers simultaneously. The difference is only in time. Now imagine how many DBAs and how many months you will need if Opatch wasn't automated. Even if you had an infinite amount of time and such a large head count at your disposal, it is unlikely the DBAs will do an equal quality job consistently. After all, we have Oracle patching best practices built into the Ansible logic.

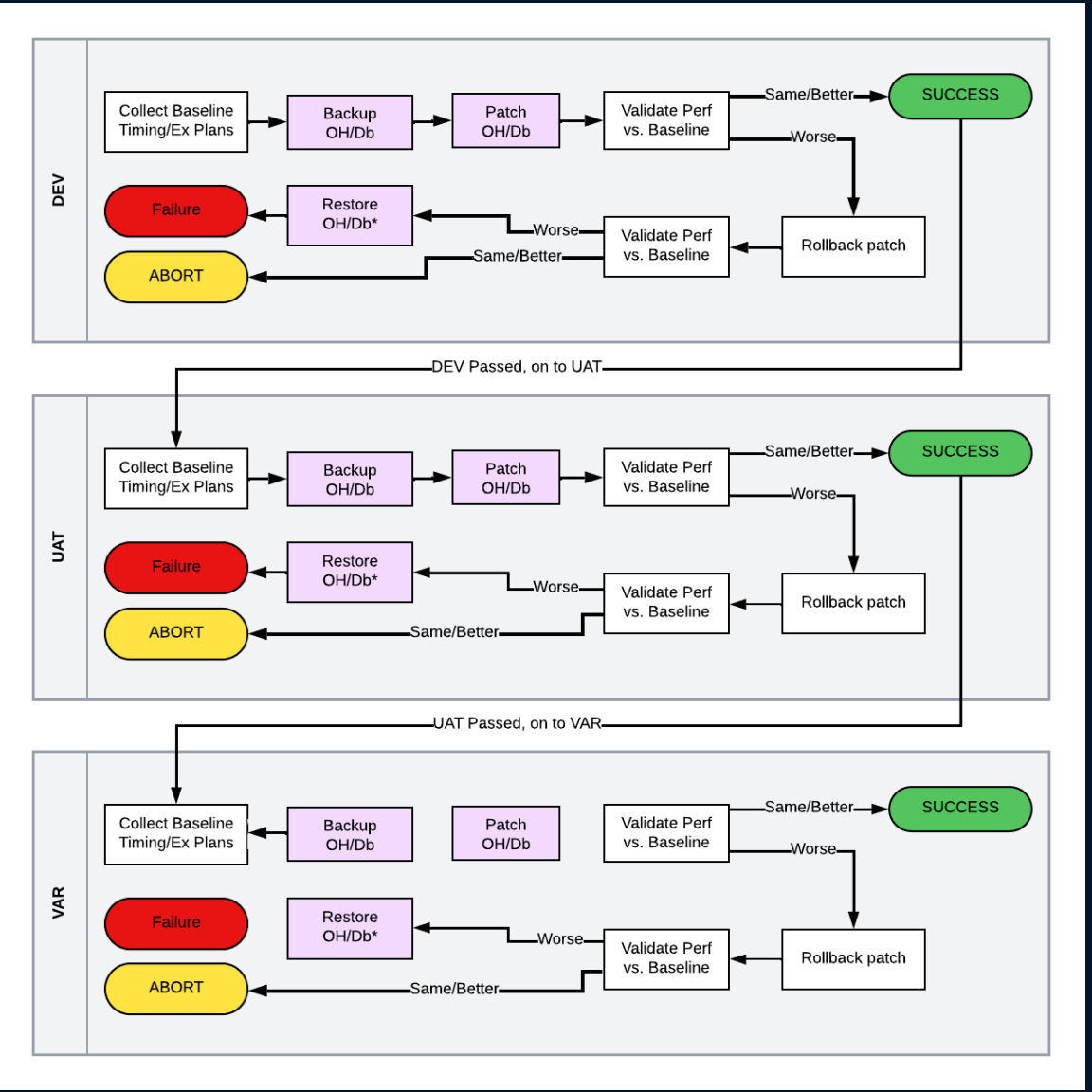
Ansible Opatch, Single Process Decision Flow



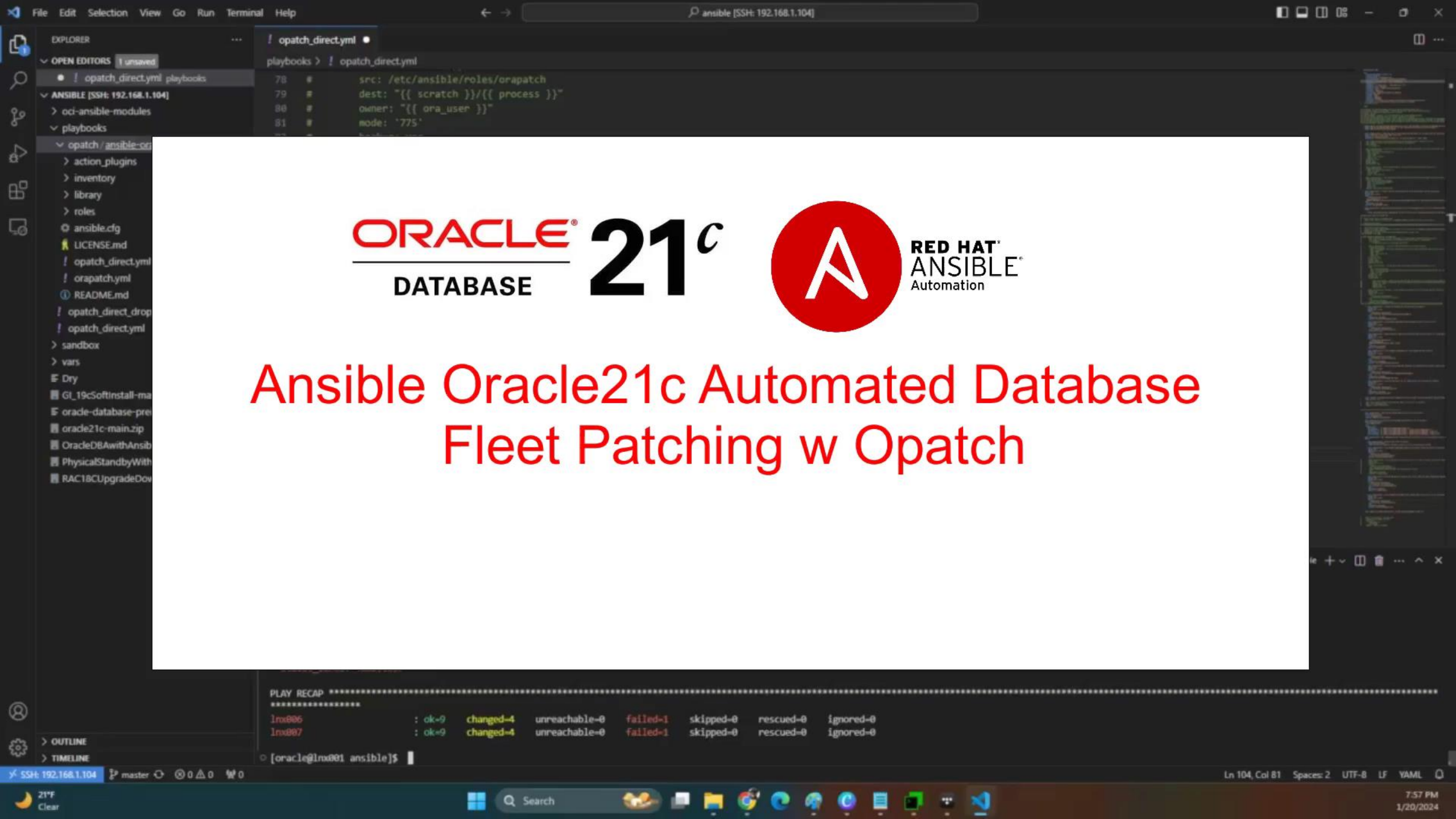
Ansible Opatch, Cluster Process Decision Flow (Option)



Ansible Opatch,
Sequential Unattended Lower Environment Auto-Patching (Option)



If the following demo doesn't open
within the PPT after pressing the "Play" button, it can be accessed online:
https://grigorian.tech/chase/


ORACLE
DATABASE
21c
RED HAT
ANSIBLE
Automation
Ansible Oracle21c Automated Database
Fleet Patching w Opatch

If the previous demo failed to open
within the PPT after pressing the "Play" button, it can be accessed online:
https://grigorian.tech/chase/

A Call to Action

Around 30% of our competition is already automating. We have the power of automation at our disposal. It can make us faster, smarter, more scalable and ready for the future challenges.

1. Lets identify the what WE can/should automate (most time spent on what? GoldenGate refreshes? Migrations? Upgrades? Deployments? DataGuard builds? Anything time consuming and labor intensive and or complex).
2. Lets automate those processes, measure success, re-improve, re-measure, till improvements aren't worth the time anymore.
3. Then move on to next process, repeat until all hard to control and time-consuming database operation processes become efficient.