ITM
Institución Universitaria
Acreditada en Alta Calidad

ANDRES
MARTINEZ
GUTIERREZ

2018-2

DATABASE ADMINISTRATION ADVANCED

AGENDA

- Introduction to Oracle
- Docker
- How to install Oracle
- ACID Principle

Introduction to Oracle

https://docs.oracle.com/cd/E11882_01/server.112/e40540/intro.htm#CNCPT001

Oracle

- Larry Ellison, Bob Miner and Ed Oates in 1977
- Written in Assembly, C, C++
- Multiplatform
- Initial Version: Oracle V2. V2.3
- 11g Release 1 Sep 2008
- 11g Release 2 Aug 2013
- 12c (12.1.0.1) Release 1 June 2013
- 12c (12.1.0.2) Release 1 July 2014
- 12c Release 2 March 2017
- 18c Feb 2018

Oracle Editions

A. 12c Enterprise Edition **(EE)**

B. 12c Standard Edition 2 **(SE2)**

C. 11g R2 Express Edition **(XE)**

   A. 1 CPU, up to 1Gb RAM, 11Gb Storing (Data)

D. Oracle Personal Edition

Oracle Platforms

A. Linux on x86-64 (only Red Hat Enterprise Linux, Oracle Linux and

B. SUSE distributions are supported)

C. Microsoft Windows on x86-64

D. Oracle Solaris on SPARC and x86-64

E. IBM AIX on POWER Systems

F. IBM Linux on z Systems

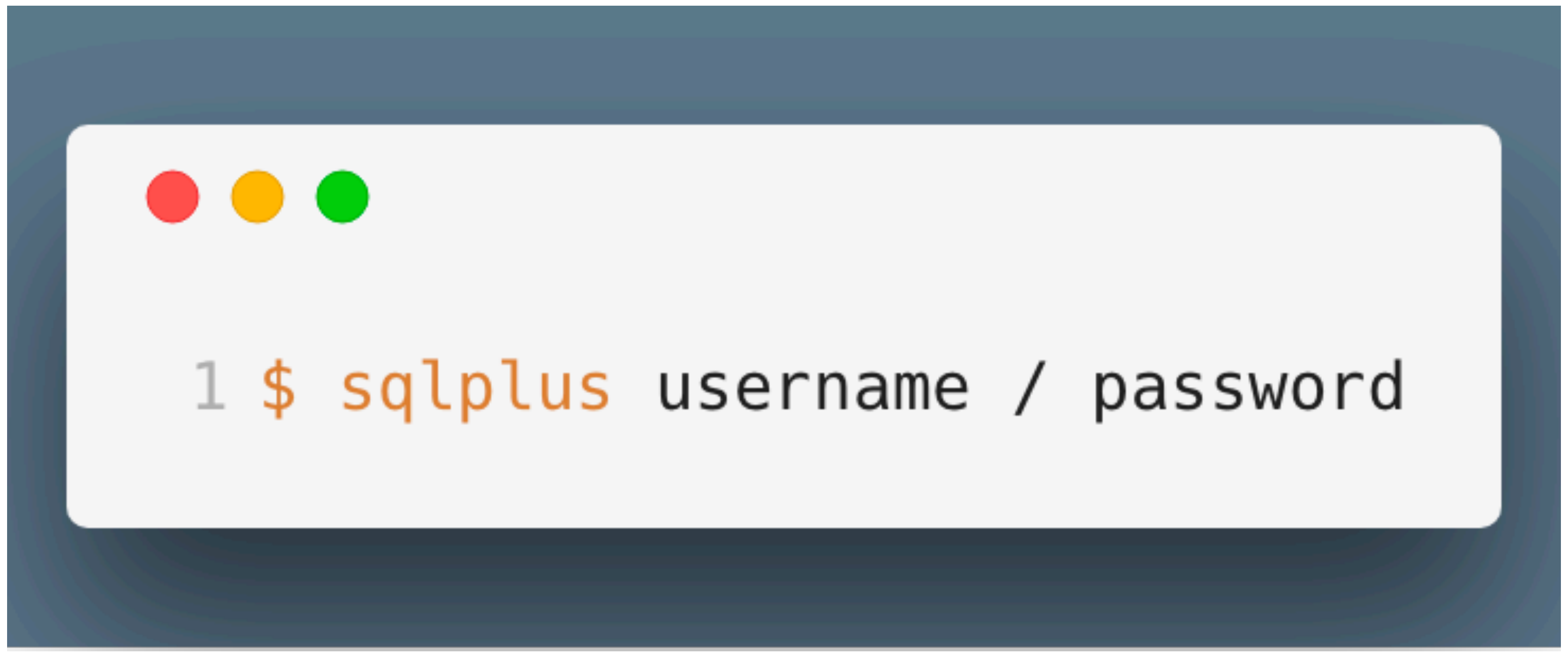
G. HP-UX on Itanium

Oracle Tools

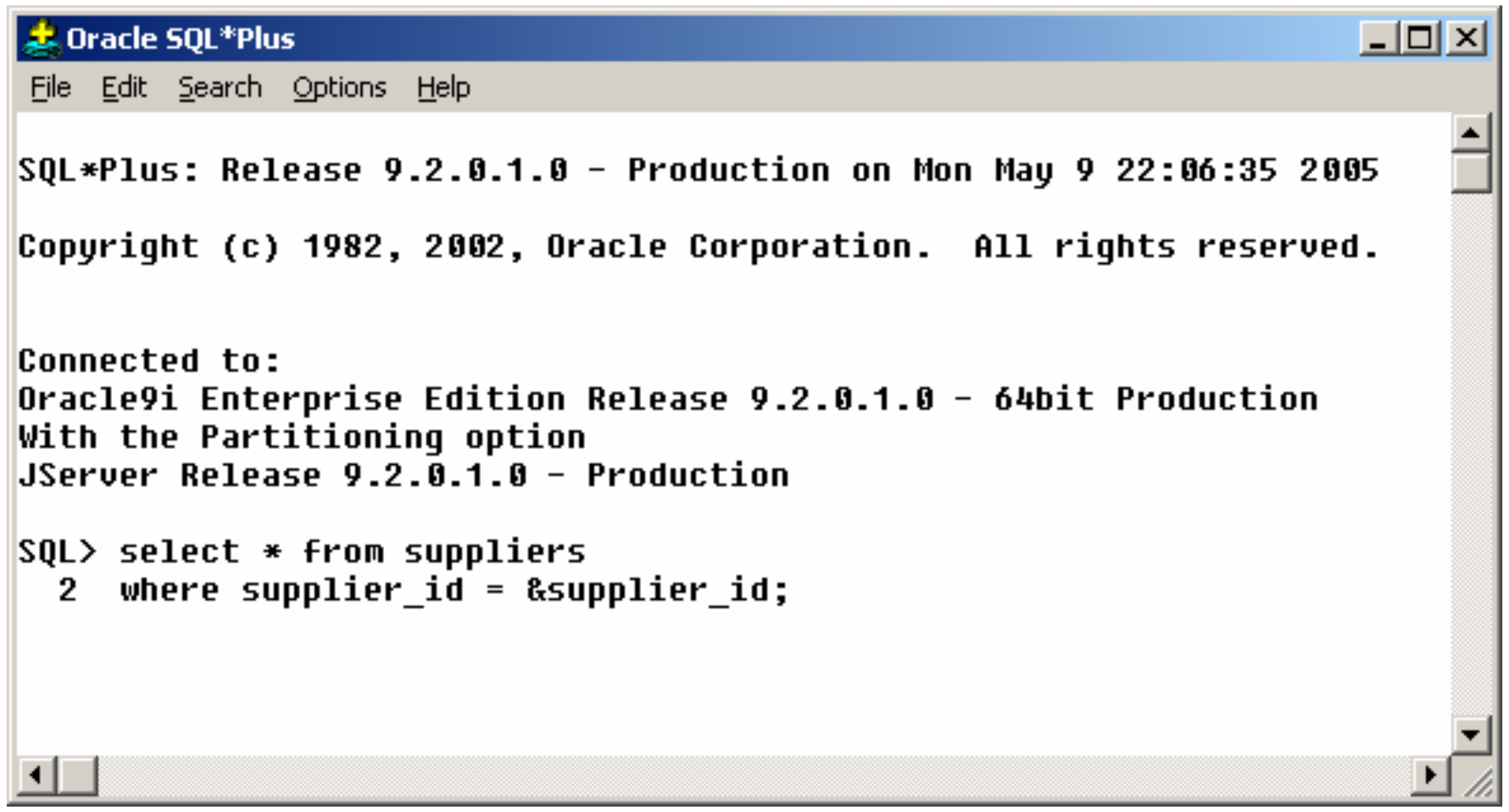
A.SQL Plus
B.TOAD
C.Sql Developer

ORACLE®

Oracle Tools / SQL Plus



```
$ sqlplus username / password
```



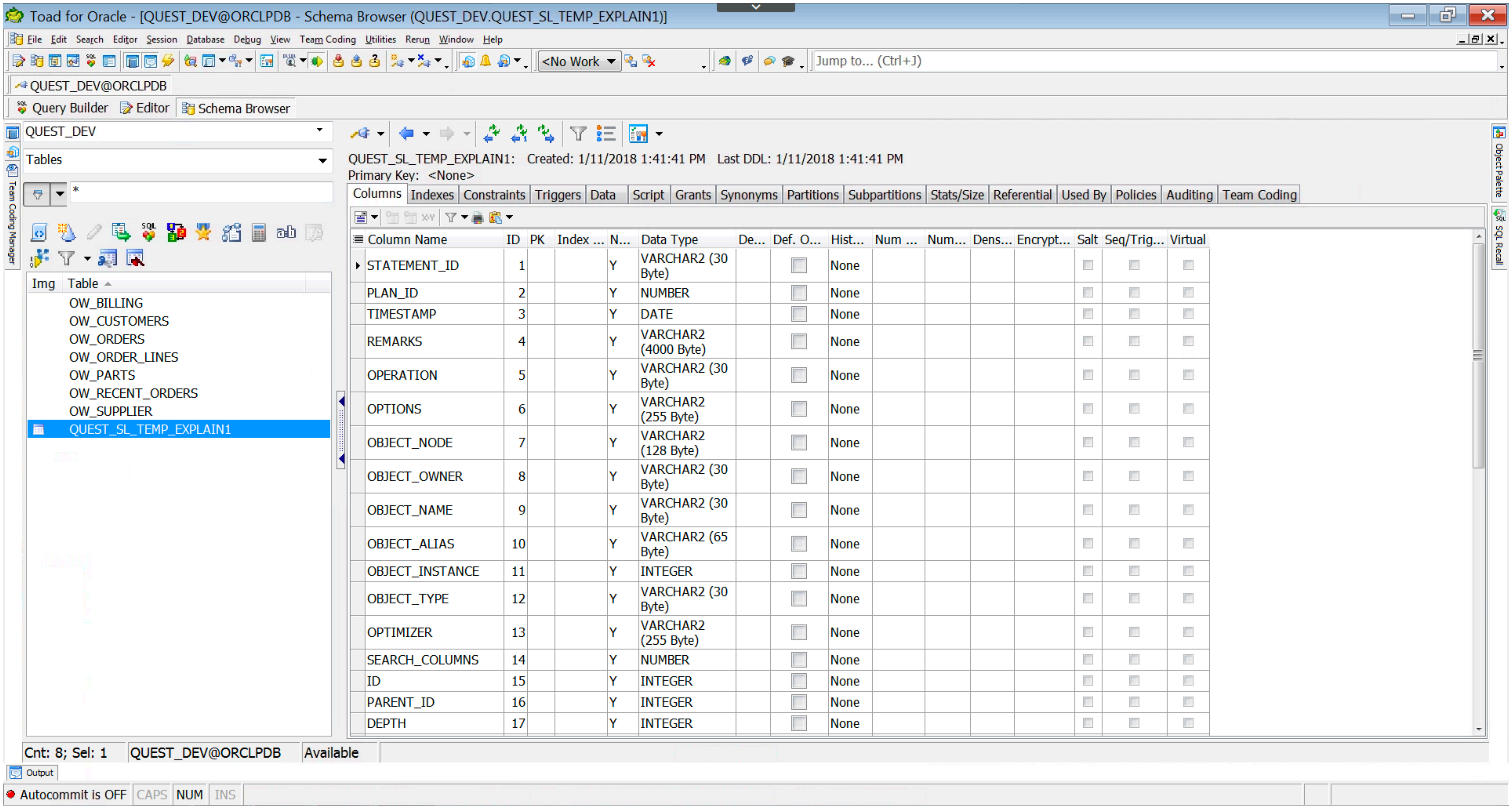
```
SQL*Plus: Release 9.2.0.1.0 - Production on Mon May 9 22:06:35 2005

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.


Connected to:
Oracle9i Enterprise Edition Release 9.2.0.1.0 - 64bit Production
With the Partitioning option
JServer Release 9.2.0.1.0 - Production

SQL> select * from suppliers
  2  where supplier_id = &supplier_id;
```

Oracle Tools / TOAD



Toad for Oracle - [QUEST_DEV@ORCLPDB - Schema Browser (QUEST_DEV.QUEST_SL_TEMP_EXPLAIN1)]

Img	Table
	OW_BILLING
	OW_CUSTOMERS
	OW_ORDERS
	OW_ORDER_LINES
	OW_PARTS
	OW_RECENT_ORDERS
	OW_SUPPLIER
	QUEST_SL_TEMP_EXPLAIN1

QUEST_SL_TEMP_EXPLAIN1: Created: 1/11/2018 1:41:41 PM Last DDL: 1/11/2018 1:41:41 PM

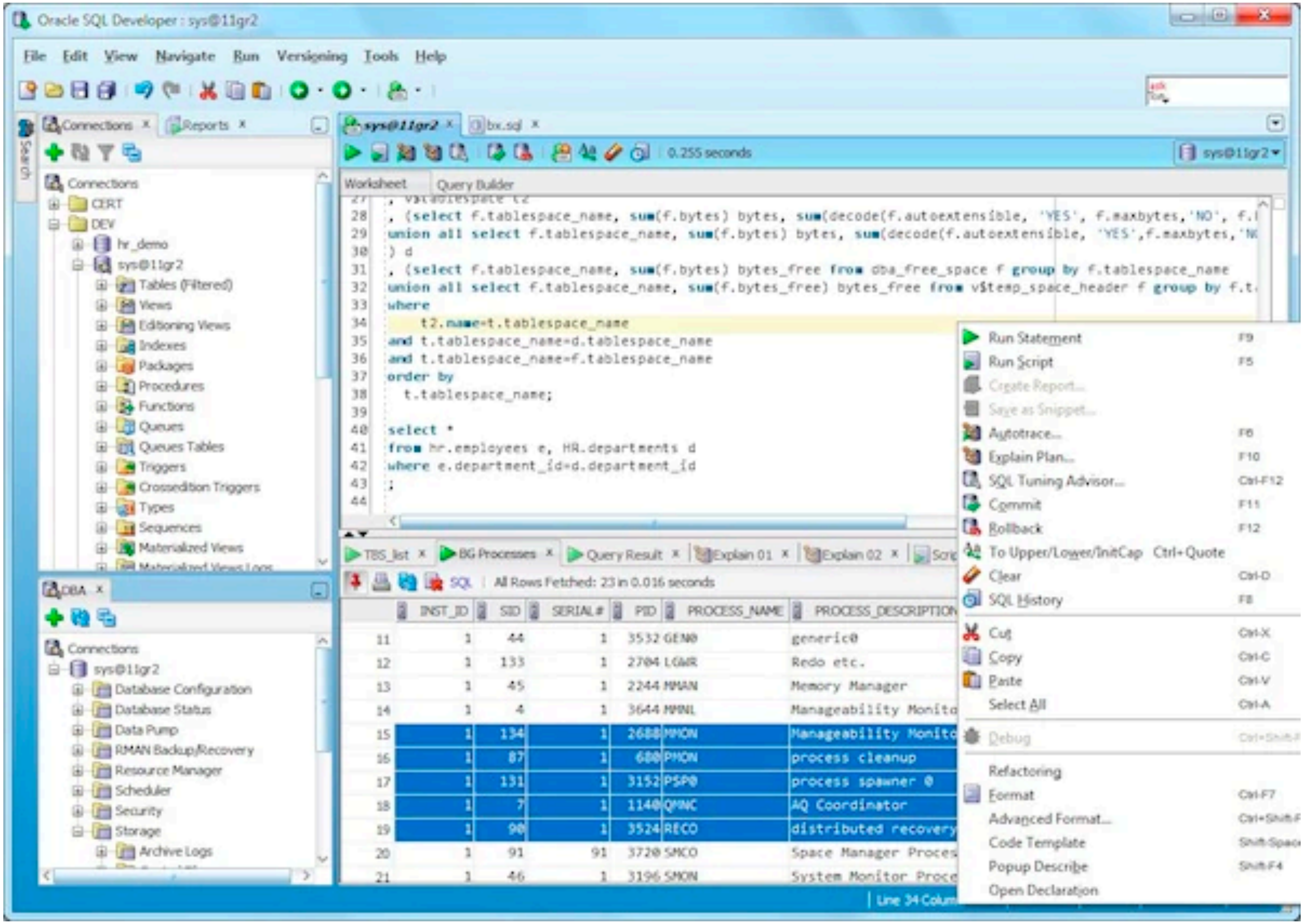
Primary Key: <None>

Column Name	ID	PK	Index ...	N...	Data Type	De...	Def. O...	Hist...	Num ...	Num...	Dens...	Encrypt...	Salt	Seq/Trig...	Virtual
STATEMENT_ID	1			Y	VARCHAR2 (30 Byte)		☐	None					☐	☐	☐
PLAN_ID	2			Y	NUMBER		☐	None					☐	☐	☐
TIMESTAMP	3			Y	DATE		☐	None					☐	☐	☐
REMARKS	4			Y	VARCHAR2 (4000 Byte)		☐	None					☐	☐	☐
OPERATION	5			Y	VARCHAR2 (30 Byte)		☐	None					☐	☐	☐
OPTIONS	6			Y	VARCHAR2 (255 Byte)		☐	None					☐	☐	☐
OBJECT_NODE	7			Y	VARCHAR2 (128 Byte)		☐	None					☐	☐	☐
OBJECT_OWNER	8			Y	VARCHAR2 (30 Byte)		☐	None					☐	☐	☐
OBJECT_NAME	9			Y	VARCHAR2 (30 Byte)		☐	None					☐	☐	☐
OBJECT_ALIAS	10			Y	VARCHAR2 (65 Byte)		☐	None					☐	☐	☐
OBJECT_INSTANCE	11			Y	INTEGER		☐	None					☐	☐	☐
OBJECT_TYPE	12			Y	VARCHAR2 (30 Byte)		☐	None					☐	☐	☐
OPTIMIZER	13			Y	VARCHAR2 (255 Byte)		☐	None					☐	☐	☐
SEARCH_COLUMNS	14			Y	NUMBER		☐	None					☐	☐	☐
ID	15			Y	INTEGER		☐	None					☐	☐	☐
PARENT_ID	16			Y	INTEGER		☐	None					☐	☐	☐
DEPTH	17			Y	INTEGER		☐	None					☐	☐	☐

Cnt: 8; Sel: 1 | QUEST_DEV@ORCLPDB | Available

● Autocommit is OFF

Oracle Tools / SqlDeveloper



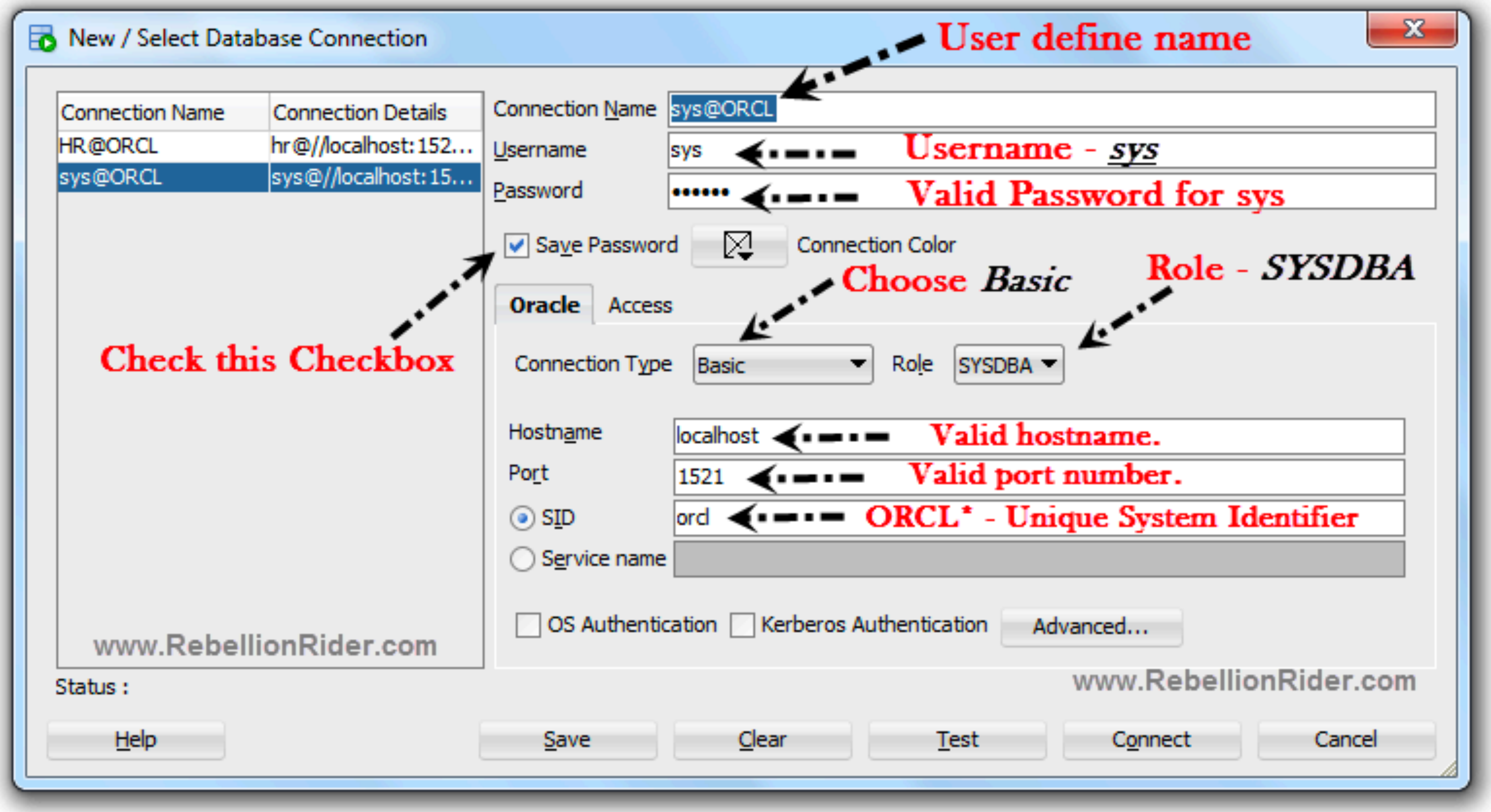
```
, v$tablespace t2
, (select f.tablespace_name, sum(f.bytes) bytes, sum(decode(f.autoextensible, 'YES', f.maxbytes,'NO', f.l
union all select f.tablespace_name, sum(f.bytes) bytes, sum(decode(f.autoextensible, 'YES',f.maxbytes,'N(
) d
, (select f.tablespace_name, sum(f.bytes) bytes_free from dba_free_space f group by f.tablespace_name
union all select f.tablespace_name, sum(f.bytes_free) bytes_free from v$temp_space_header f group by f.t
where
    t2.name=t.tablespace_name
and t.tablespace_name=d.tablespace_name
and t.tablespace_name=f.tablespace_name
order by
  t.tablespace_name;

select *
from hr.employees e, HR.departments d
where e.department_id=d.department_id
;
```

All Rows Fetched: 23 in 0.016 seconds

	INST_ID	SID	SERIAL#	PID	PROCESS_NAME	PROCESS_DESCRIPTION
11	1	44	1	3532	GEN0	generic0
12	1	133	1	2704	LGWR	Redo etc.
13	1	45	1	2244	MMAN	Memory Manager
14	1	4	1	3644	MMNL	Manageability Monito
15	1	134	1	2688	MMON	Manageability Monito
16	1	87	1	680	PMON	process cleanup
17	1	131	1	3152	PSP0	process spawner 0
18	1	7	1	1140	QMNC	AQ Coordinator
19	1	90	1	3524	RECO	distributed recovery
20	1	91	91	3720	SMCO	Space Manager Proces
21	1	46	1	3196	SMON	System Monitor Proce

Oracle Tools / SqlDeveloper



New / Select Database Connection

Connection Name	Connection Details
HR@ORCL	hr@//localhost:152...
sys@ORCL	sys@//localhost:15...

Connection Name: sys@ORCL ← User define name

Username: sys ← Username - *sys*

Password: •••••• ← Valid Password for sys

☑ Save Password ← Check this Checkbox

Connection Color

Oracle | Access

Connection Type: Basic ← Choose *Basic*

Role: SYSDBA ← Role - *SYSDBA*

Hostname: localhost ← Valid hostname.

Port: 1521 ← Valid port number.

● SID: orcl ← ORCL* - Unique System Identifier

○ Service name:

□ OS Authentication □ Kerberos Authentication

Status :

DOCKER

https://www.toptal.com/devops/getting-started-with-docker-simplifying-devops

Docker

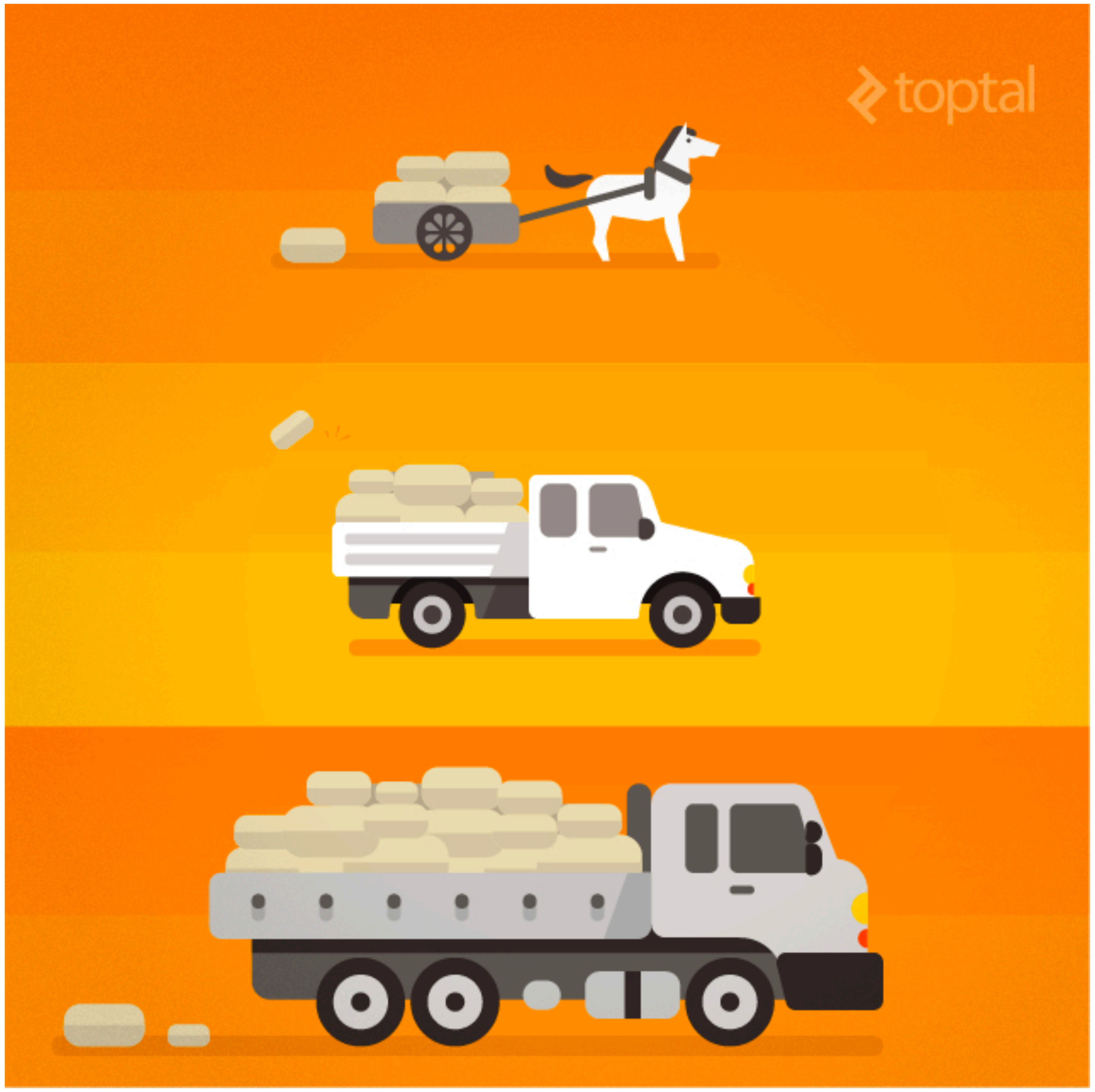
1. Open source project that facilitates deployment of applications
2. inside of software containers.
3. First version: 2013
4. Motto: Build - Ship - Run
   1. **Build**: compose your application from micro-services
   2. **Ship**: design the entire cycle of application development, testing, and distribution
   3. **Run**: deploy scalable services securely and reliably on a wide variety of platforms.

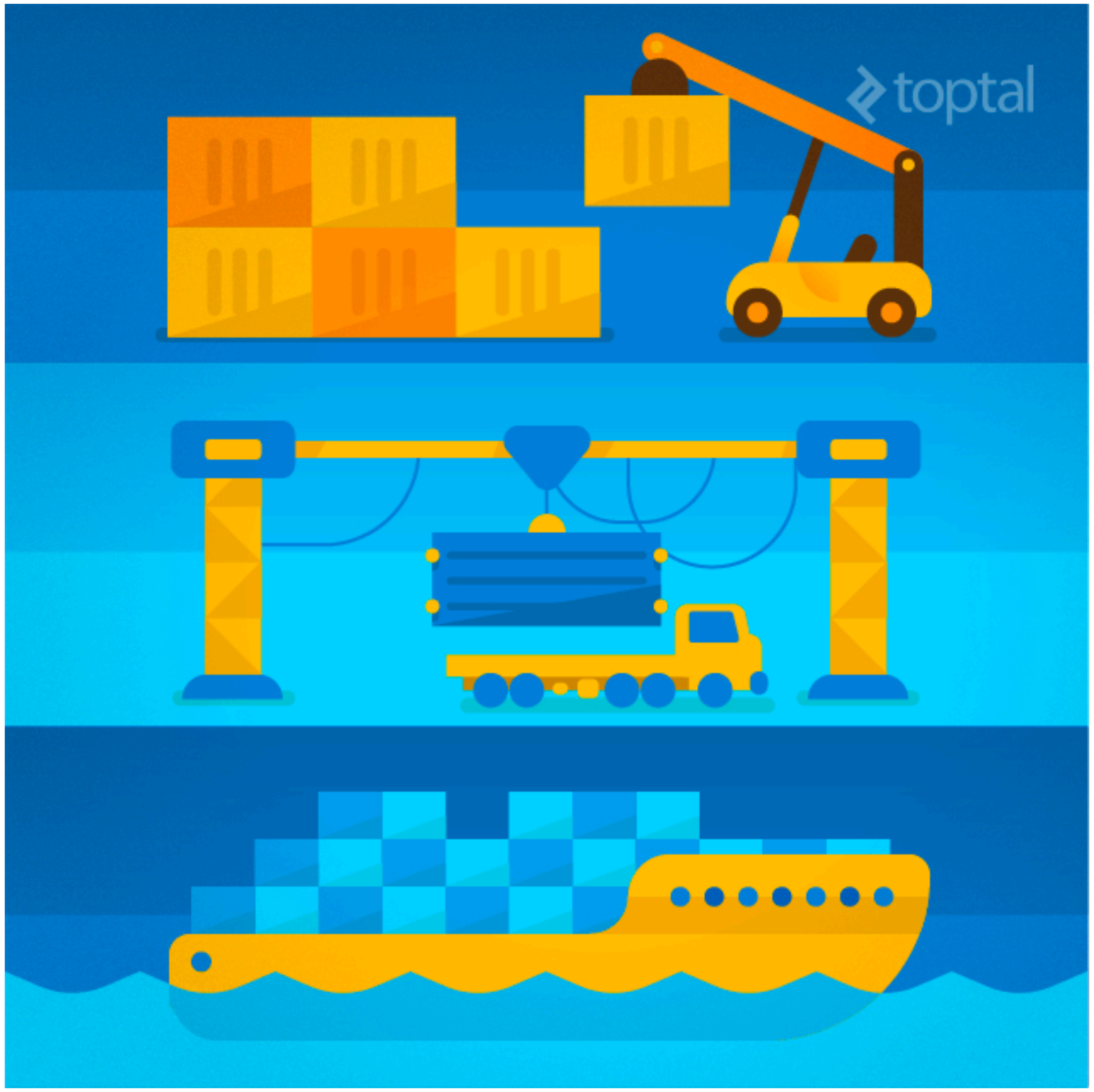
Install Docker

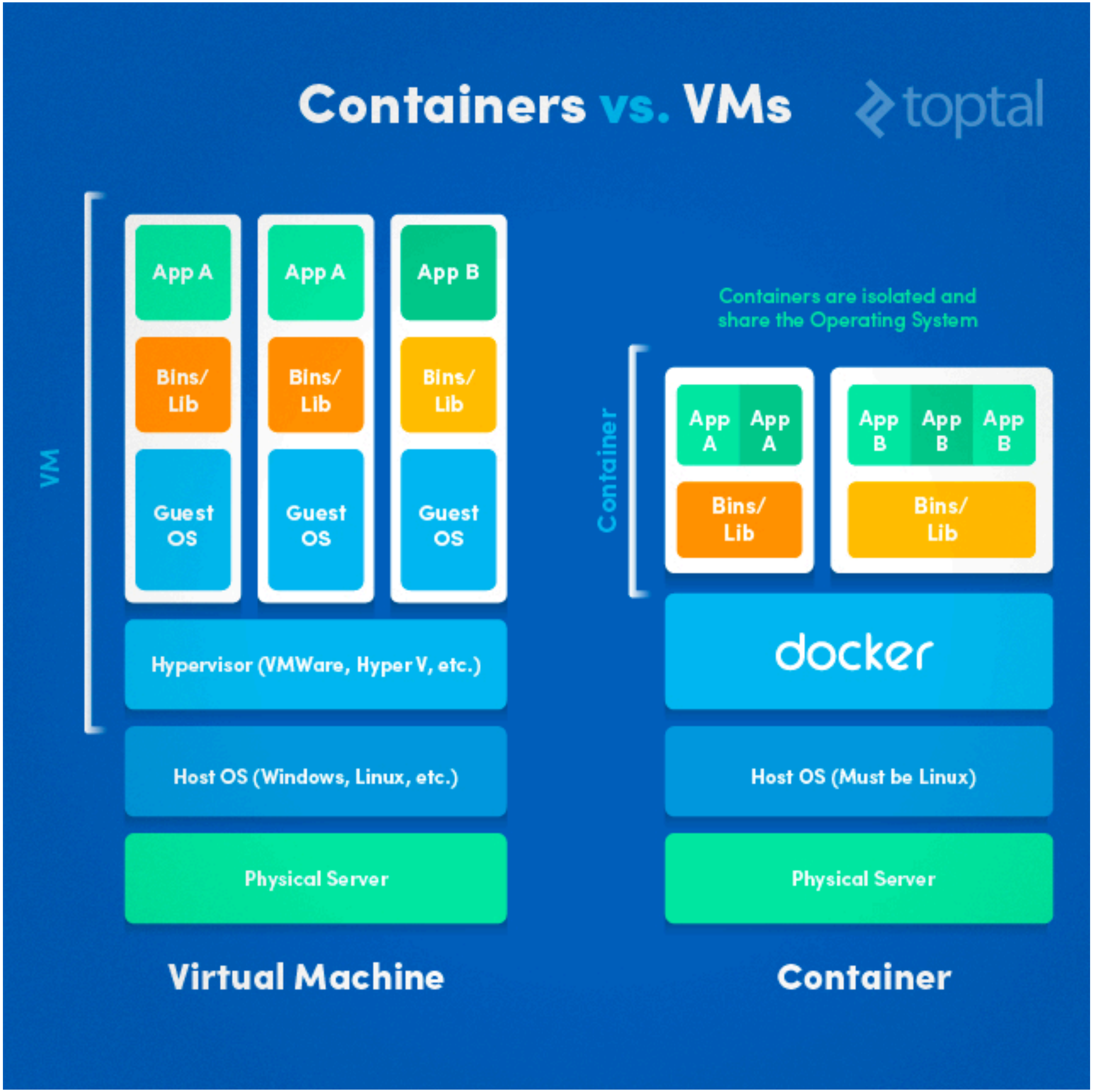
1. There are two editions: Community Edition (CE) and Enterprise Edition (EE)

    1. **CE**: Developers and small teams (stable)
    2. **EE**: Enterprise development and IT teams

3. https://docs.docker.com/install/


toptal


toptal

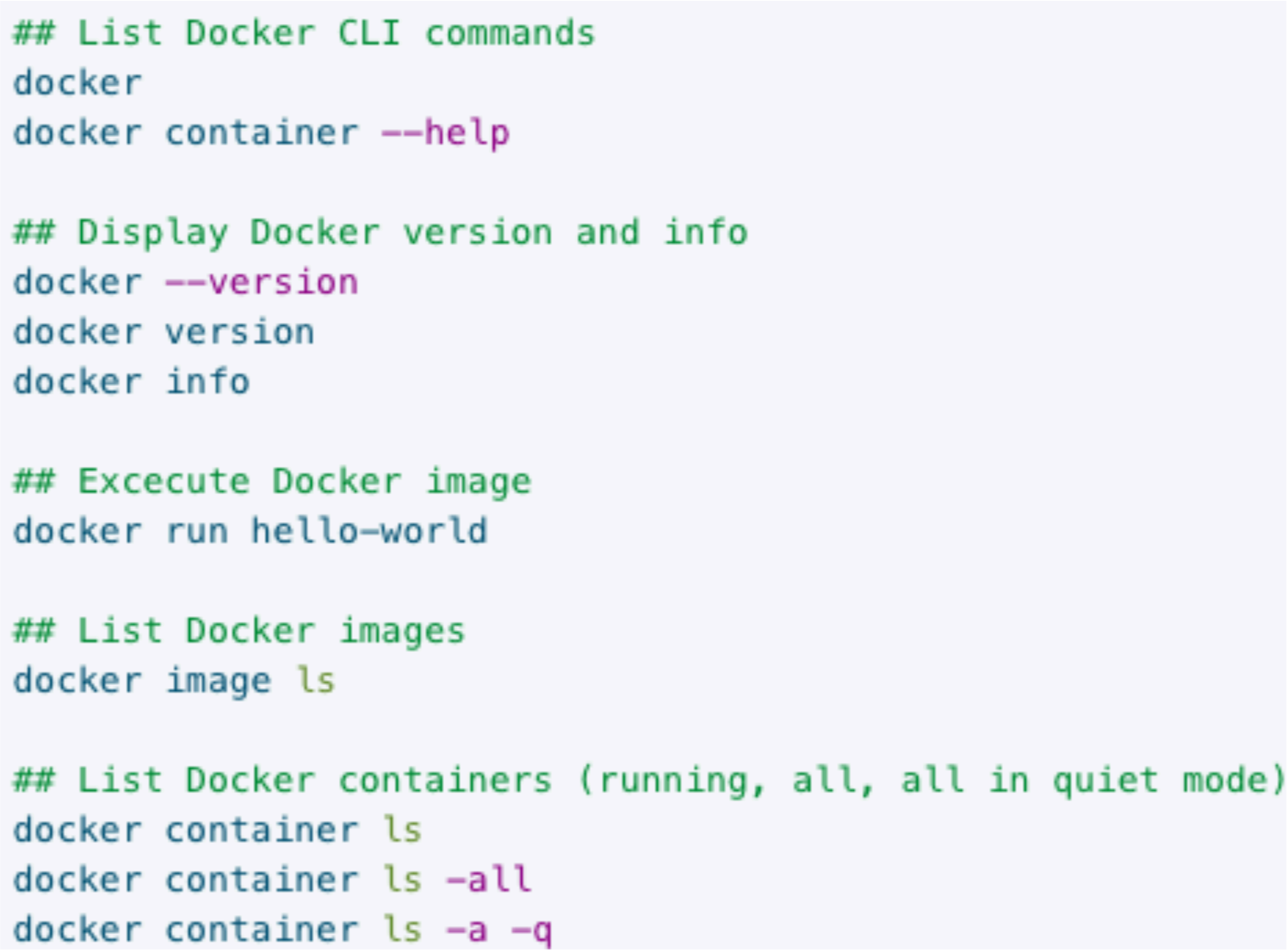

Containers vs. VMs
toptal
VM
App A
App A
App B
Bins/ Lib
Bins/ Lib
Bins/ Lib
Guest OS
Guest OS
Guest OS
Hypervisor (VMWare, Hyper V, etc.)
Host OS (Windows, Linux, etc.)
Physical Server
Virtual Machine
Containers are isolated and share the Operating System
Container
App A
App A
App B
App B
App B
Bins/ Lib
Bins/ Lib
docker
Host OS (Must be Linux)
Physical Server
Container

Docker Images - Containers

- A **container** is launched by running an image

- An **image** is an executable package that includes everything needed to run an application–the code, a runtime, libraries, environment variables, and configuration files

- A **container** is a runtime instance of an image–what the image becomes in memory when executed



```
## List Docker CLI commands
docker
docker container --help

## Display Docker version and info
docker --version
docker version
docker info

## Excecute Docker image
docker run hello-world

## List Docker images
docker image ls

## List Docker containers (running, all, all in quiet mode)
docker container ls
docker container ls -all
docker container ls -a -q
```

How to Install Oracle?

How to Install Oracle?

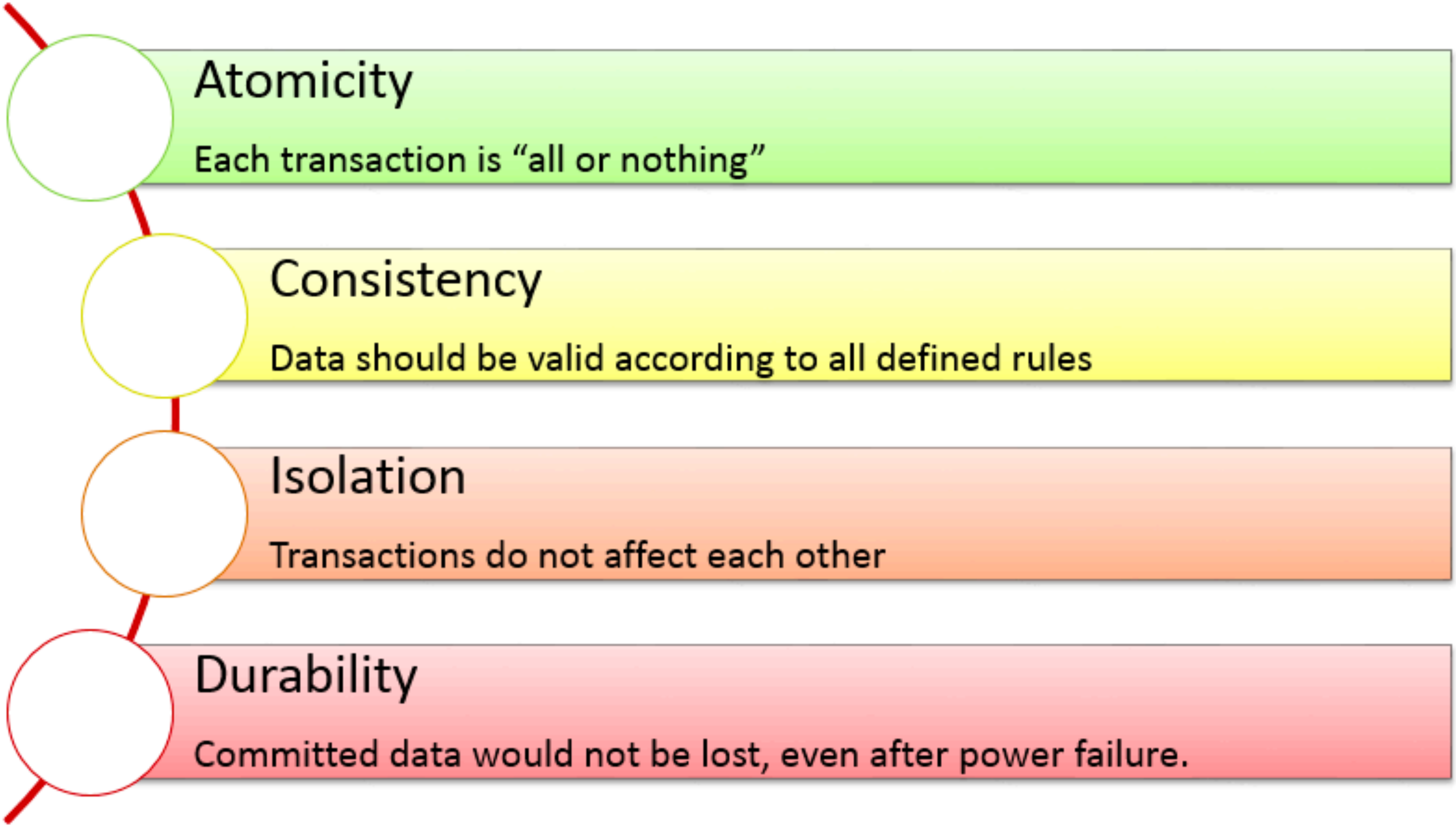
- **Install as usual**: Download files, install step by step:
  - https://youtu.be/juLD9T5GAck

- **Install through Docker (Recommended)**
  - **Oracle 11g:** https://hub.docker.com/r/wnameless/oracle-xe-11g/
  - **Oracle 12c:** https://hub.docker.com/r/sath89/oracle-12c/

Don’t forget to install Sql Developer or TOAD

ACID Principle

ACID Properties



Atomicity

Each transaction is “all or nothing”

Consistency

Data should be valid according to all defined rules

Isolation

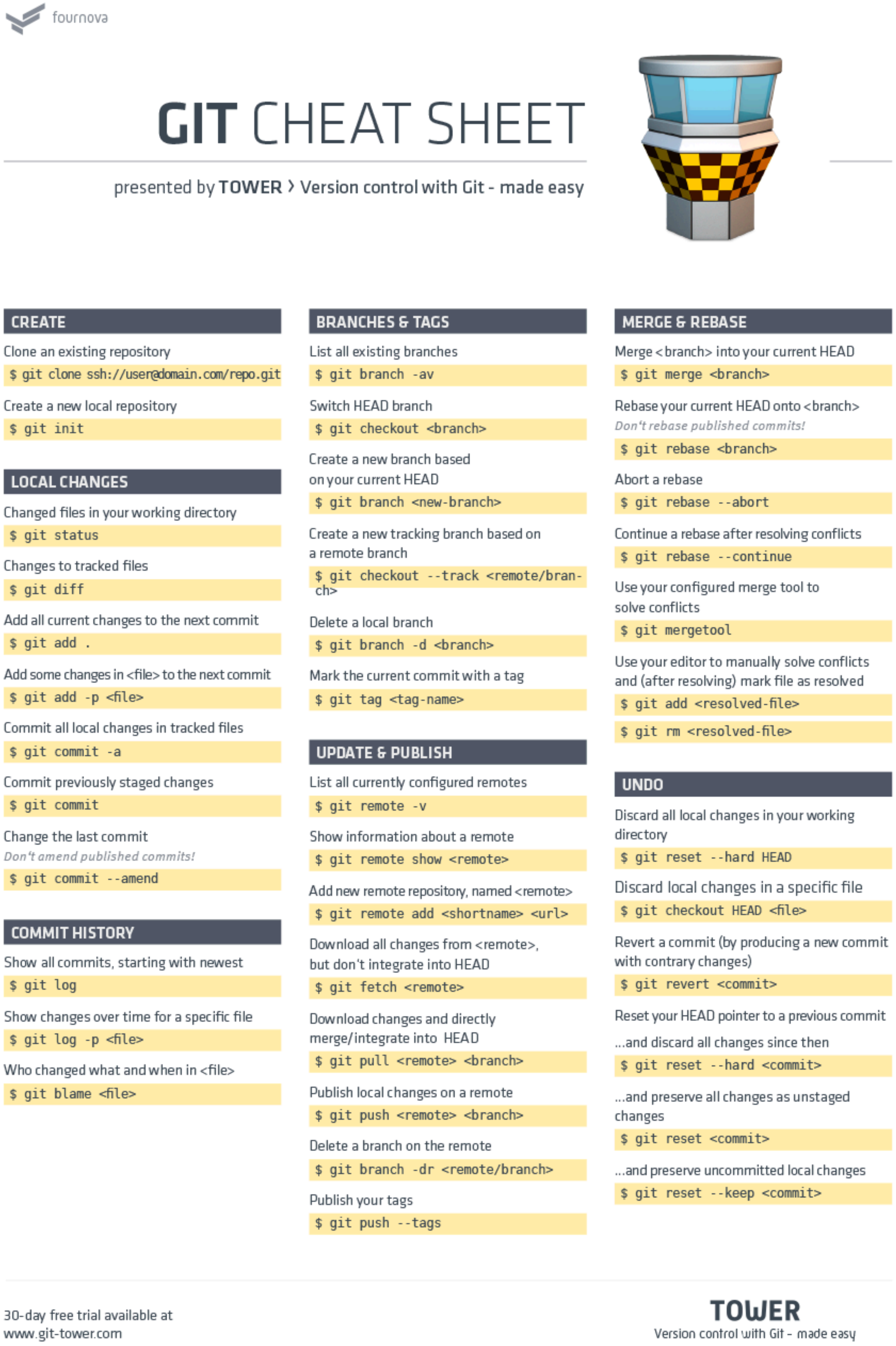
Transactions do not affect each other

Durability

Committed data would not be lost, even after power failure.

Git Cheatsheet

https://www.git-tower.com/blog/git-cheat-sheet/



fournova

GIT CHEAT SHEET

presented by **TOWER** › Version control with Git - made easy

CREATE

Clone an existing repository

```
$ git clone ssh://user@domain.com/repo.git
```

Create a new local repository

```
$ git init
```

LOCAL CHANGES

Changed files in your working directory

```
$ git status
```

Changes to tracked files

```
$ git diff
```

Add all current changes to the next commit

```
$ git add .
```

Add some changes in <file> to the next commit

```
$ git add -p <file>
```

Commit all local changes in tracked files

```
$ git commit -a
```

Commit previously staged changes

```
$ git commit
```

Change the last commit

Don't amend published commits!

```
$ git commit --amend
```

COMMIT HISTORY

Show all commits, starting with newest

```
$ git log
```

Show changes over time for a specific file

```
$ git log -p <file>
```

Who changed what and when in <file>

```
$ git blame <file>
```

BRANCHES & TAGS

List all existing branches

```
$ git branch -av
```

Switch HEAD branch

```
$ git checkout <branch>
```

Create a new branch based on your current HEAD

```
$ git branch <new-branch>
```

Create a new tracking branch based on a remote branch

```
$ git checkout --track <remote/bran-
ch>
```

Delete a local branch

```
$ git branch -d <branch>
```

Mark the current commit with a tag

```
$ git tag <tag-name>
```

UPDATE & PUBLISH

List all currently configured remotes

```
$ git remote -v
```

Show information about a remote

```
$ git remote show <remote>
```

Add new remote repository, named <remote>

```
$ git remote add <shortname> <url>
```

Download all changes from <remote>, but don't integrate into HEAD

```
$ git fetch <remote>
```

Download changes and directly merge/integrate into HEAD

```
$ git pull <remote> <branch>
```

Publish local changes on a remote

```
$ git push <remote> <branch>
```

Delete a branch on the remote

```
$ git branch -dr <remote/branch>
```

Publish your tags

```
$ git push --tags
```

MERGE & REBASE

Merge <branch> into your current HEAD

```
$ git merge <branch>
```

Rebase your current HEAD onto <branch>

Don't rebase published commits!

```
$ git rebase <branch>
```

Abort a rebase

```
$ git rebase --abort
```

Continue a rebase after resolving conflicts

```
$ git rebase --continue
```

Use your configured merge tool to solve conflicts

```
$ git mergetool
```

Use your editor to manually solve conflicts and (after resolving) mark file as resolved

```
$ git add <resolved-file>
```

```
$ git rm <resolved-file>
```

UNDO

Discard all local changes in your working directory

```
$ git reset --hard HEAD
```

Discard local changes in a specific file

```
$ git checkout HEAD <file>
```

Revert a commit (by producing a new commit with contrary changes)

```
$ git revert <commit>
```

Reset your HEAD pointer to a previous commit

...and discard all changes since then

```
$ git reset --hard <commit>
```

...and preserve all changes as unstaged changes

```
$ git reset <commit>
```

...and preserve uncommitted local changes

```
$ git reset --keep <commit>
```

30-day free trial available at
www.git-tower.com

TOWER
Version control with Git - made easy

Thank you!