
elephant-shed Documentation

Release 1.0

credativ

Dec 22, 2017

Contents

1	Intro	1
2	Installation	3
2.1	Package Installation	3
2.2	Installation from source	4
3	First Steps	7
4	Components	11
4.1	PostgreSQL	11
4.2	Portal	12
4.3	Service Web Interface - Cockpit	15
4.4	Monitoring - Prometheus	15
4.5	Dashboard - Grafana	17
4.6	DBA Tool - pgAdmin4	20
4.7	Backup - pgBackRest	20
4.8	Reporting - pgBadger	25
4.9	Web Terminal - Shell In A Box	26
4.10	Firewall - ferm	26
4.11	Remote Control - tmate	27
4.12	Configuration Revision - etckeeper	28
5	Users	29
6	Limitations	31
7	Known Bugs and Issues	33
7.1	PostgreSQL	33
7.2	pgadmin4	33
7.3	Portal	33
7.4	Cockpit	33
8	License	35
9	Support and more	37
9.1	Do you have any question or want to know more?	37
9.2	Do you need professional support or additional services?	37

CHAPTER 1

Intro

Elephant Shed is a web-based PostgreSQL management front-end that bundles several utilities and applications for use with PostgreSQL. It currently manages single-node Debian/Ubuntu PostgreSQL servers and appliances.

The main components are:

- PostgreSQL - <https://www.postgresql.org/>
- pgAdmin4 - <https://www.pgadmin.org/>
- postgresql-common - <https://anonscm.debian.org/cgit/pkg-postgresql/postgresql-common.git>
- pgBadger - <http://dalibo.github.io/pgbadger/>
- pgBackRest - <http://www.pgbackrest.org/>
- Grafana - <https://grafana.com/>
- Prometheus - <https://prometheus.io/>
- Cockpit - <http://cockpit-project.org/>
- Shell In A Box - <https://github.com/shellinabox/shellinabox>

In addition several other tools are included for configuration management and setup.

The number of components bundled and tasks handled add some overhead compared to running just the database server. It is therefore only recommended for adequately sized systems.

This document describes the current version. Updated versions of this document will be shipped with the elephant-shed packages and can be found in `/usr/share/doc/elephant-shed-portal` (`/usr/share/doc/elephant-shed*`) and in the web portal under <https://your-server/doc/>.

Elephant Shed consists of the following Debian packages and their dependencies:

- `elephant-shed`: Metapackage that includes the following packages.
- `elephant-shed-prometheus`: Configuration files and helper scripts for Prometheus and its exporters.
- `elephant-shed-cockpit`: Configuration files for cockpit and cockpit-ws.
- `elephant-shed-grafana`: Preconfigured Prometheus datasource and dashboard that includes various system and PostgreSQL metrics.
- `elephant-shed-pgadmin4`: Configuration files for pgAdmin4.
- `elephant-shed-pgbackrest`: Systemd service files and generators, helper scripts and preset configuration.
- `elephant-shed-pgbadger`: Systemd service files, generators and helper scripts.
- `elephant-shed-portal`: Elephant Shed web portal including Apache configuration.
- `elephant-shed-postgresql`: Additional preset configuration files for PostgreSQL.
- `elephant-shed-shellinabox`: Shell In A Box configuration files.
- `elephant-shed-tmate`: Preconfigured tmate installation for easier support.

2.1 Package Installation

Prebuilt packages are available from <https://packages.credativ.com/public/postgresql/>.

```
# Install tools
sudo apt-get install curl ca-certificates apt-transport-https

# Use official PostgreSQL repo, apt.postgresql.org
echo "deb http://apt.postgresql.org/pub/repos/apt/ stretch-pgdg main" | sudo tee -a /
→etc/apt/sources.list.d/pgdg.list
curl https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
```

```
# Use credativ repo, packages.credativ.com
echo "deb https://packages.credativ.com/public/postgresql/ stretch-stable main" |_
↪sudo tee -a /etc/apt/sources.list.d/elephant-shed.list
curl https://packages.credativ.com/public/postgresql/aptly.key | sudo apt-key add -

# Install elephant-shed
sudo apt-get update
sudo apt-get install elephant-shed

# Choose desired PostgreSQL versions to install
sudo apt-get install postgresql-10

# Every user in the group "elephant-shed" is allowed to login at the portal
# Add all needed users to this group
sudo adduser <USERNAME> elephant-shed
```

The repository also contains packages that the `elephant-shed` packages depend on. This includes packages for Grafana, Cockpit and various python libraries.

2.2 Installation from source

The source code is available on GitHub: <https://github.com/credativ/elephant-shed>

2.2.1 Build Debian Packages

All Elephant Shed Debian packages can be built using the command `make deb`.

Requirements:

- `dpkg-dev`
- `devscripts`
- `jq`

2.2.2 Build Documentation

To build the documentation in HTML as well as the PDF format type `make README`.

Requirements:

- `pandoc`
- `texlive-latex-base`

2.2.3 Create Testsystem with Vagrant

The `make vagrant` command builds all components, creates a new virtual machine using Vagrant and deploys the software using Ansible. This can also be used to redeploy a already running machine.

The Vagrant configuration is located in `vagrant/Vagrantfile`.

Requirements:

- `vagrant`
- `virtualbox` or `libvirt`
- `ansible`

2.2.4 Deploy on remote machine

To deploy the software on any machine, enter the connection information in the inventory `vagrant/inventory`. The deployment can then be started with the following command `make ansible`.

Requirements:

- `ansible`

CHAPTER 3

First Steps

Log into your web browser and go to the server's IP address (e.g. <https://your-server/>). The default setup will redirect HTTP requests to HTTPS.

The Elephant Shed portal provides information about running PostgreSQL instances and their status. Moreover you get access to all other installed components.

The server will ask you for your user credentials. Depending on the deployment process the required user will differ. On a test installation (e.g. using Vagrant) the initial user is **admin** with password **admin**. See also: [Users](#). All bundled components except for pgAdmin4 have been configured to use PAM authentication.

pgAdmin4 doesn't support PAM authentication yet. It has its own user management system which is decoupled from all system users. You should have been asked about the initial user within the installation setup. If you deployed a Vagrant VM the initial user is **admin@localhost** and **admin** as password.

PostgreSQL Appliance Dashboard

Logged in as *admin* - Logout

PostgreSQL Cluster

Cluster	Port	Data directory	Archiving	Full Backup	Incr Backup	
● 9.6/main	5432	/mnt/pgdata/9.6/main	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	▲
<div> <div>systemd</div> <div>Report</div> <div>Backup</div> </div> <div> <div>Service</div> <div>Log</div> <div>•</div> <div>Run</div> <div>Show</div> <div>•</div> <div>Full</div> <div>Incremental</div> <div>Info</div> </div>						
● 10/main	5433	/mnt/pgdata/10/main	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	▼
● 10/test	5434	/mnt/pgdata/10/test	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	▼

On a new installation you will find one cluster running the current PostgreSQL major version with the name `main`.

The configuration for clusters can be found in `/etc/postgresql/<major version>/<name>/`.

To use PostgreSQL from external application servers only a few steps are needed.

1. Open a shell connection to the server using SSH or shellinabox <https://your-server/shellinabox>.
2. Switch to user `postgres` and launch `psql`:
 - `sudo -u postgres psql`
3. Create a database and corresponding application user, options:
 - `psql: CREATE ROLE appuser1 WITH LOGIN PASSWORD 'testpass';`
 - `psql: CREATE DATABASE appdb1 OWNER appuser1;`
4. Allow external access for your application servers, your network or everyone. Configuration file: `/etc/postgresql/<major version>/<name>/pg_hba.conf`
5. (optional) Make desired configuration changes and tuning. `/etc/postgresql/<major version>/<name>/postgresql.conf`
6. Reload the configuration, options:
 - Portal: Click on the button `Service` next to the cluster and choose "Reload" from the dropdown menu

- psql: `SELECT pg_reload_conf();`

7. (optional) Configure a superuser to be able to use pgAdmin4 <https://your-server/pgadmin4> or other management tools

- Create password for user postgres: `\password`
- Create personalized superusers: `CREATE USER "sosna" SUPERUSER; \password "sosna"`

4.1 PostgreSQL

The Elephant Shed is based on `postgresql-common`, the default PostgreSQL management system for Debian based installations. Tasks like creating, dropping or renaming a PostgreSQL instance ("cluster" in PostgreSQL terms) should be done through `postgresql-common`'s command line utilities `pg_createcluster`, `pg_dropcluster` and `pg_renamecluster`.

4.1.1 Default Configuration

Beside the `postgresql-common` default configuration the `elephant-shed-postgresql` package adds additional parameters for the cluster creation process. Some of these parameters are required by the Elephant Shed components. E.g. `pgBadger` requires some special `log_line_prefix`. You can find this configuration under `/etc/postgresql-common/createcluster.d/elephant-shed.conf`. Be careful when changing any of these values.

4.1.2 Cluster Administration

To create a new cluster issue the command `pg_createcluster <version> <name>`. Installed clusters and their status can be listed via `pg_lsclusters`.

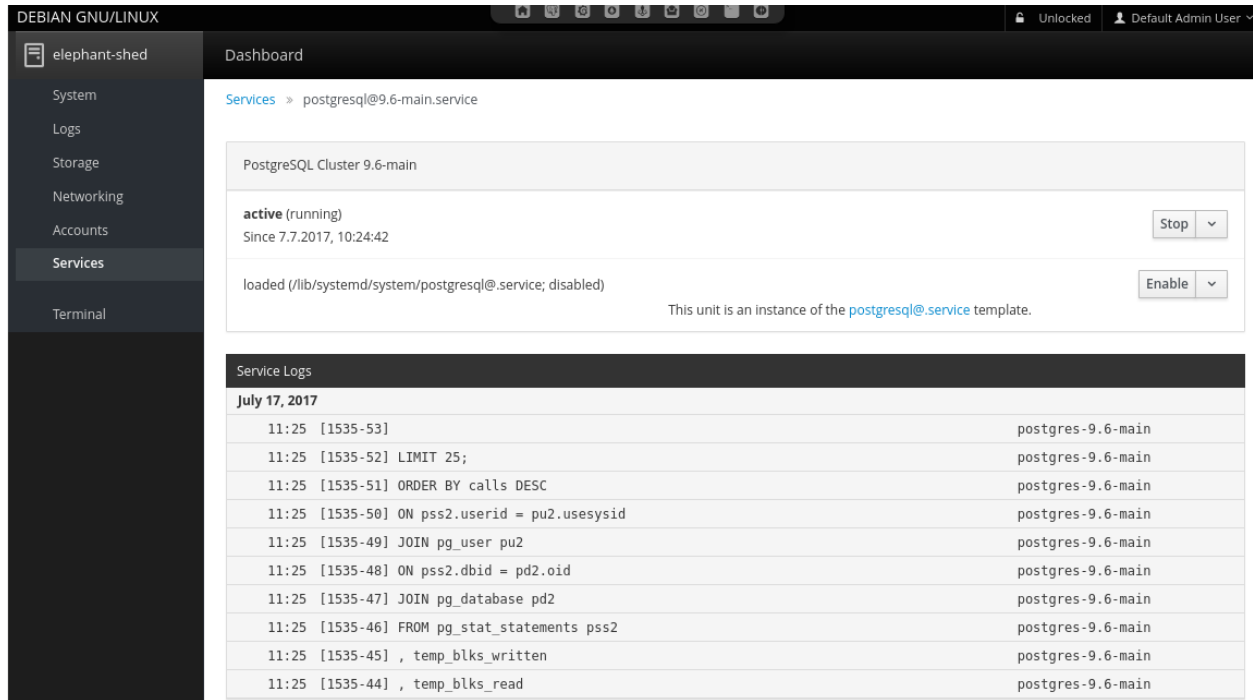
```
postgres@stretch:~$ pg_lsclusters
Ver Cluster  Port Status           Owner    Data directory  Log file
9.6 main      5432 online             postgres /9.6/main        /log/9.6-main.log
9.6 standby1  5433 online,recovery postgres /9.6/standby1    /log/9.6-standby1.log
9.6 standby2  5434 online,recovery postgres /9.6/standby2    /log/9.6-standby2.log
9.6 test      5435 online             postgres /9.6/test         /log/9.6-test.log
```

To delete a cluster use `pg_dropcluster <version> <name>` (be careful, this removes all data in the cluster as well!).

To start, stop, restart or reload use `pg_ctlcluster <action> <version> <name>` with the following commands as action:

- start
- stop
- restart
- reload

Alternatively, you can use `systemctl` (`systemctl <action> postgresql@<version>-<name>`) or the Cockpit web interface.



Note: Future versions of The Elephant Shed will include cluster management in the *portal*.

4.2 Portal

The portal serves as an entry point to all other web-based interfaces. It uses HTTPS and basic authentication. Each user within the Unix group `elephant-shed` has access to it (see *Users*).













The portal also shows the status of all PostgreSQL cluster including links to the Cockpit service (in order to start or stop the cluster), the log files, pgBadger reports and the backup software pgBackRest.

A navigation bar at the top allows switching between the different web services.

By default only a self signed certificate for HTTPS is deployed. A corresponding security warning is shown in most browsers. You can change the certificate with a signed one (e.g. from your company CA, or from [Let's Encrypt](#)). The web services are served by Apache2. It also acts as a reverse proxy to serve all other web interfaces and to enforce authentication.

4.2.1 PostgreSQL Cluster

PostgreSQL Cluster

Cluster	Port	Data directory	Archiving	Full Backup	Incr Backup
 9.6/main	5432	/mnt/pgdata/9.6/main			
<div style="display: flex; justify-content: space-around;"> <div> systemd Mgmt Log • Run Show </div> <div> Report • Full Incr Info </div> </div>					
 9.6/standby	5433	/mnt/pgdata/9.6/standby			
<div style="display: flex; justify-content: space-around;"> <div> systemd Mgmt Log • Run </div> <div> Report • Full Incr </div> </div>					
 9.6/testing	5434	/mnt/pgdata/9.6/testing			
<div style="display: flex; justify-content: space-around;"> <div> systemd Mgmt Log • Run </div> <div> Report • Full Incr </div> </div>					

This section presents an overview of the existing PostgreSQL clusters and their status. For each cluster, a set of switches shows the current status. By clicking on a cluster an additional menu with buttons opens. Currently all buttons link to the corresponding components where a confirmation is required so no actions are triggered directly, but this may change in the future.

4.2.2 systemd - Service

This links to the configuration of this PostgreSQL cluster in Cockpit. Here it is possible to configure the corresponding service to be enabled or disabled on system start and also trigger actions like start, stop and reload.

4.2.3 systemd - Log

Links to the corresponding log entries in Cockpit if syslog is enabled for this cluster (which is the default for clusters created by Elephant Shed).

4.2.4 Report - Run

By default pgBadger reports for all clusters are generated once every night. With this service it is possible to generate a report for a specific cluster on demand.

4.2.5 Report - Show

Links to the corresponding pgBadger report overview. See *pgBadger* for more information.

4.2.6 Backup

This sections provides several functions for backups using pgBackRest. For more information about the backup tool pgBackRest visit *pgBackRest*.

Full

Link to Cockpit for starting an ad-hoc full backup.

Incremental

Link to Cockpit for starting an ad-hoc incremental backup.

Info

Shows the status of the backups. This button is only shown after the first backup run. Here the available backups and the content of the WAL archive is shown.

Additional information can be found here: <http://www.pgbackrest.org/user-guide.html#quickstart/backup-info>

4.2.7 Switches

Archiving

This switch shows if an archive command is set that uses pgBackRest. It is possible to set one or to deactivate the feature by using `'/bin/true'`. Archiving is needed for point in time recovery but more importantly for pgBackRest backups. Archiving will be activated automatic if a backup is triggered via the portal or timers.

To change it manually the service `pgbackrest-toggle-archiving@<version>-<name>.service` can be started. This toggles the state.

Full Backup

Switch to enable or disable a periodical backup. An enabled backup job (systemd timer) is shown by green color. To start/stop the timer `pgbackrest@<version>-<name>.timer` needs to be started or stopped. Enable/disable is used to enable/disable the timer after the next reboot.

Incr Backup

Switch to enable or disable a periodical backup. An enabled backup job (systemd timer) is shown by green color. To start/stop the timer `pgbackrest-incr@<version>-<name>.timer` needs to be started or stopped. Enable/disable is used to enable/disable the timer after the next reboot.

4.3 Service Web Interface - Cockpit

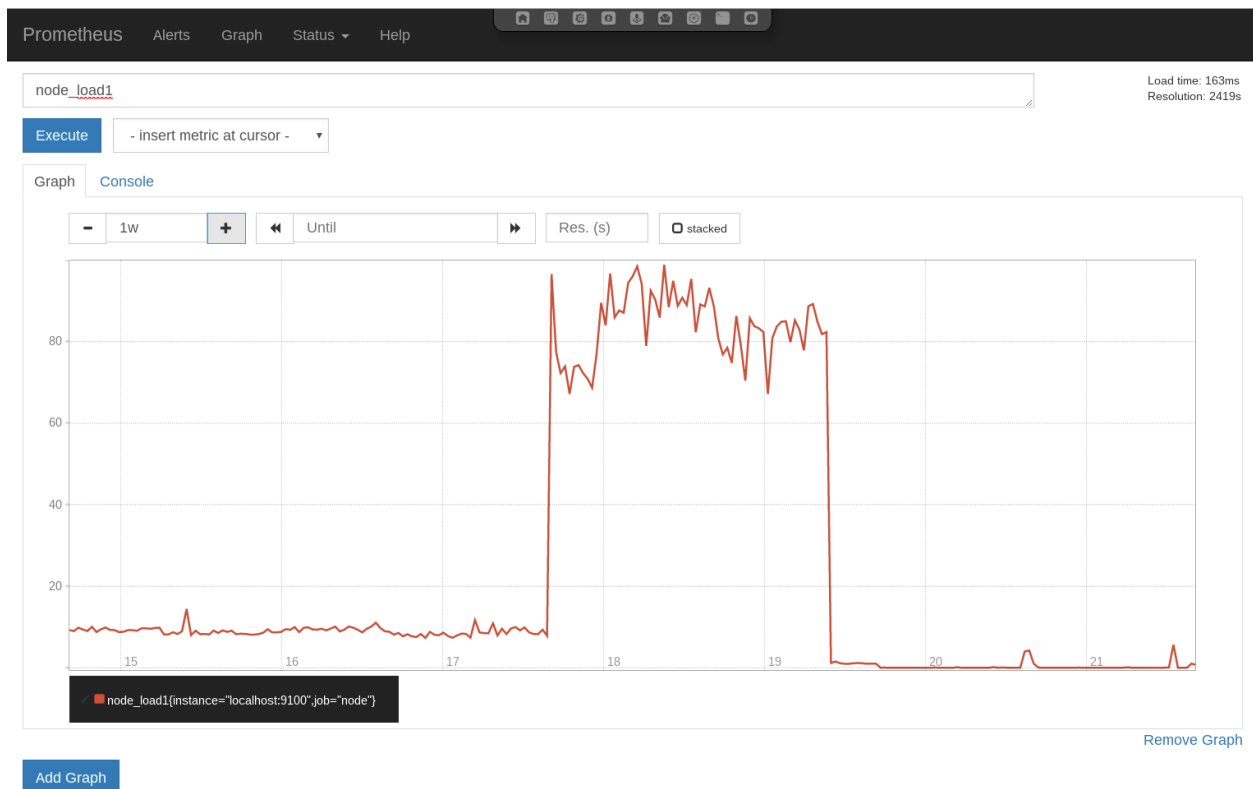
Cockpit allows remote management of all systemd related services via HTTPS. It makes starting, stopping or restarting of services as simple as clicking a button. It also shows system logs in real-time.

The screenshot shows the Cockpit web interface for a Debian GNU/Linux stretch system. The top navigation bar includes a sidebar with links to System, Logs, Storage, Networking, Accounts, Services, Software Updates (selected), and Terminal. The main content area displays the 'Dashboard' with a summary of 229 updates, including 39 security fixes. A 'Check for updates' button is present. Below this, a section titled 'Available Updates' features a warning box stating that Cockpit itself will be updated. A table lists the available updates:

Name	Version	Bugs	Details
apache2, apache2-bin, apache2-data, apache2-utils	2.4.27-2	851094 , 868467	<p>Security Update: CVE-2017-3167, CVE-2017-3169, CVE-2017-7659, CVE-2017-7668, CVE-2017-7679, CVE-2017-9788</p> <p>* Switch back to openssl 1.0 for now. The transition to 1.1 needs more work and should go into experimental, first. Reopens: #851094 -- Stefan Fritsch <sf@debian.org> Sun, 16 Jul 2017 23:01:10 +0200</p> <p>== 2.4.27-1 == [New upstream release] * Fix CVE-2017-9788: mod_auth_digest: Uninitialized memory reflection Closes: #868467 [Stefan Fritsch] * Switch to openssl 1.1. Closes: #851094 -- Stefan Fritsch <sf@debian.org> Sun, 16 Jul 2017 10:39:15 +0200</p> <p>== 2.4.25-4 == * Backport security fixes from 2.4.26: * CVE-2017-3167: Authentication bypass with ap_get_basic_auth_pw() * CVE-2017-3169: mod_ssl NULL pointer dereference * CVE-2017-7668: Buffer overrun in ap_find_token() * CVE-2017-7679: mod_mime buffer overread * CVE-2017-7659: mod_http2 NULL pointer dereference -- Stefan Fritsch <sf@debian.org> Tue, 20 Jun 2017 21:31:51 +0200</p>
fontconfig-config, libfontconfig1	2.12.3-0.2	756715 , 799416 , 816045 , 862483	<p>Security Update: CVE-2016-5384</p> <p>* Non-maintainer upload. * Add a NEWS file to describe the change in the default hinting style. Also</p>

4.4 Monitoring - Prometheus

Prometheus is a metric based monitoring system for servers and services. It collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results, and can trigger alerts if some condition is observed to be true.



Targets				
node				
Endpoint	State	Labels	Last Scrape	Error
http://localhost:9100/metrics	UP	instance="localhost:9100"	7.304s ago	
prometheus				
Endpoint	State	Labels	Last Scrape	Error
http://localhost:9090/prometheus/metrics	UP	instance="localhost:9090"	25.994s ago	
sql_exporter				
Endpoint	State	Labels	Last Scrape	Error
http://localhost:9237/metrics	UP	instance="localhost:9237"	53.028s ago	

4.4.1 Services

In this setup the Prometheus stack consists of different components controlled by systemd. The following units are deployed.

prometheus.service

Monitoring system and time series database - This is the monitoring service itself. It actively pulls the metrics from the different sources. It also provides internal metrics and a web interface which is accessible via the portal.

Configuration files:

- /etc/prometheus/elephant-shed-prometheus.yml

- `/etc/default/elephant-shed-prometheus`

prometheus-node-exporter.service

Prometheus exporter for machine metrics - This service exports the system metrics and listens on port 9100. These metrics are collected every 30 seconds by default.

Configuration files:

- `/etc/default/elephant-shed-prometheus-node-exporter`

prometheus-sql-exporter.service

Prometheus exporter for SQL metrics - This service collects the PostgreSQL specific metrics and listens on port 9237. The metrics are retrieved by querying the database. In order to not to generate additional load the metrics are collected only every 60 seconds.

WARNING: It is not advisable to set the monitoring interval for the `prometheus-sql-exporter` lower then 60 Seconds. This could interference with normal applications and has a high impact on the PostgreSQL cluster.

The `prometheus-sql-exporter.service` starts one connection to each database on startup and keeps this connection open. At the beginning of each connection the `prometheus-sql-exporter.service` checks if the extension `pg_stat_statements` is present. If not, the service issues the statement `CREATE EXTENSION pg_stat_statements`.

Configuration files:

- `/etc/prometheus-sql-exporter.yml`

update-prometheus-sql-exporter-config.timer

This timer triggers the `update-prometheus-sql-exporter-config.service` periodically which generates a new configuration for the `prometheus-sql-exporter` every 10 minutes. This makes sure that every new database cluster and every new database is included in the monitoring automatically. It's possible to call the `update-prometheus-sql-exporter-config.service` manually to generate a new configuration directly.

Configuration template file:

- `prometheus-sql-exporter.yml.in`

Configuration file (runtime):

- `prometheus-sql-exporter.yml`

4.4.2 Additional Resources

- <https://prometheus.io/docs/introduction/overview/>
- <https://github.com/prometheus/prometheus>

4.5 Dashboard - Grafana

Grafana is a tool to create graphs and dashboards from a variety of different data sources. A PostgreSQL Server Overview dashboard is included in the default installation to get an overview over the most needed and many helpful metrics to manage and debug PostgreSQL servers.

These pre-deployed dashboards are shipped via the `elephant-shed-graphana` Debian package and can change in the future. They are read only and need to be saved under a new name if you do any changes.

4.5.1 PostgreSQL Server Overview



This dashboard starts with a summary section with simple gauges to provide a overview of the whole system. These gauges may indicate current problems or give a hint on problems that might occur in the future.

After the gauges in-depth metrics are shown as graphs.

4.5.2 Server metrics

Server metrics include e.g.: CPU usage (by type / by core), disk usage, disk utilization, network throughput, and much more. The following template filter are configured:

- `Disk`: filter one or more disks

- Interface: filter on or more interface
- Filesystem: filter on or more filesystem / mountpoint

4.5.3 Cluster metrics

The next section contains PostgreSQL Cluster wide metrics like connections (by type / by database), number of transactions, database growth and more. Only one cluster is shown by a time. To switch the current shown cluster use the template filter `PostgreSQL Cluster`.

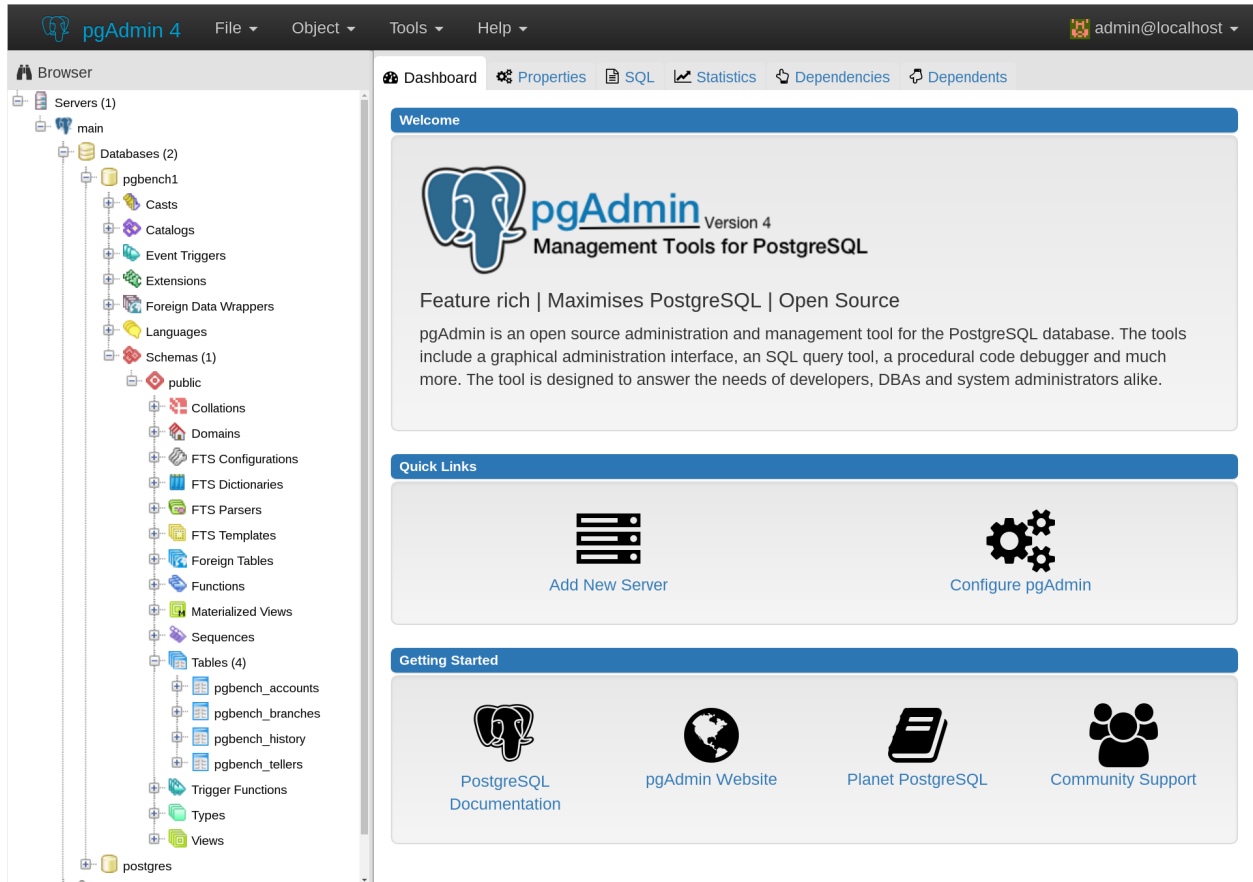
4.5.4 Database metrics

Database level metrics are shown at the end of *PostgreSQL Overview* dashboard. By default metrics for all databases of the current selected PostgreSQL cluster are shown. To filter one or more databases the template filter `Database` could be used.

4.5.5 Additional Resources

- <http://docs.grafana.org/>
- <https://github.com/grafana/grafana>

4.6 DBA Tool - pgAdmin4



pgAdmin4 is a management tool for PostgreSQL to help DBAs execute many different tasks. It provides user management, DDL functionality, an interactive SQL shell, and more.

4.6.1 Additional Resources

- <https://www.pgadmin.org/docs/pgadmin4/1.x/>

4.7 Backup - pgBackRest

The Elephant Shed comes with a preinstalled backup solution, *pgBackRest*. Each PostgreSQL instance can be backed up individually by issuing the command `systemctl start pgbackrest@<version>-<name>` or initiating a backup via Cockpit in the web interface. A shortcut is listed for each instance.

Configuration entries for each cluster are created with the first backup run. By default only `db-path` and `db-port` are set.

A list of all backups can be obtained by clicking on the pgBackRest icon on the portal site.

DEBIAN GNU/LINUX

stretch

System

Logs

Storage

Networking

Accounts

Services

Terminal

Dashboard

Services > pgbackrest@9.6-main.service

Backup PostgreSQL cluster 9.6-main using pgBackRest

inactive (dead)

loaded (/lib/systemd/system/pgbackrest@.service; static)

This unit is an instance of the [pgbackrest@.service](#) template.

Start

Mask

Service Logs

July 10, 2017

18:37	Started Backup PostgreSQL cluster 9.6-main using pgBackRest.	systemd
18:37	2017-07-10 16:37:30.207 P00 INFO: expire command end: completed successfully	pgbackrest@9.6-main
18:37	2017-07-10 16:37:30.207 P00 INFO: option 'retention-archive' is not set - archive logs will not b...	pgbackrest@9.6-main
18:37	2017-07-10 16:37:30.202 P00 INFO: expire command begin 1.20: --log-level-console=info --repo-path...	pgbackrest@9.6-main
18:37	2017-07-10 16:37:30.201 P00 INFO: backup command end: completed successfully	pgbackrest@9.6-main
18:37	2017-07-10 16:37:29.838 P00 INFO: new backup label = 20170710-163709F	pgbackrest@9.6-main
18:37	2017-07-10 16:37:29.138 P00 INFO: backup stop archive = 000000010000000000000000A, lsn = 0/A000130	pgbackrest@9.6-main
18:37	2017-07-10 16:37:26.007 P00 INFO: execute non-exclusive pg_stop_backup() and wait for all WAL seg...	pgbackrest@9.6-main
18:37	2017-07-10 16:37:26.007 P00 INFO: full backup size = 177.5MB	pgbackrest@9.6-main
18:37	2017-07-10 16:37:25.995 P01 INFO: backup file /var/lib/postgresql/9.6/main/base/1/12246 (0B, 100%)	pgbackrest@9.6-main

pgBackRest knows 3 types of backups full, incremental and differential. We are using full and differential by default. A service file for differential backups is not installed by default.

4.7.1 Full Backup

Full backups represent an complete backup of the database at a given point in time. A backup consists of two parts, the backup itself, stored in `backup` and the WAL files which were written during the backup, stored in `archive`.

To ensure these WAL files are in the archive we automatically enable WAL archiving before the first backup is created.

Warning: If archiving is enabled all WAL files newer than the oldest stored backups are kept as well. This can consume a lot space in the backup location if backups are kept for a long time and archiving is not disabled. By deleting a backup the no longer needed WAL files are removed as well.

4.7.2 Incremental Backups

Incremental backups represent the changed data between a previous full backup and the current data at a given point. Incremental backups can be significant smaller than full backups but depend on a specific previous full backup. Without this full backup they can not be restored.

4.7.3 Retention

To clean up space old backups needs to be deleted. pgBackRest needs to know how many full backups to keep. If the number is reached all additional backups will be deleted starting with the oldest. If a full backup is deleted all incremental backups depending on it will be deleted as well. This is necessary because an incremental backup can not be restored without the matching full backup.

4.7.4 Configuration

The configuration file can be found in `/etc/pgbackrest.conf`.

```
[global]
repo-path=/var/lib/pgbackrest

[9.6-main]
db-path=/var/lib/postgresql/9.6/main
db-port=5432

[9.6-test]
db-port=5433
db-path=/var/lib/postgresql/9.6/test
```

The global part sets the default configuration for every existing and future database cluster. For each single cluster theses defaults can be changed. Some basic options will be explained here. Please see the documentation for a full overview.

If the server is setup using ansible, additionally the following [global] parameters are set:

```
[global]
retention-full=4
compress-level=6
spool-path=/mnt/backup/pgbackrest_spool
archive-async=y
archive-queue-max=1099511627776
repo-path=/mnt/backup/pgbackrest
...
```

retention-full

This option defines how many full backups should be kept.

Danger: If more full backups are stored than `retention-full` pgBackRest will delete the oldest backups to keep exactly `retention-full` full backups!

compress-level

The gzip compression level to use (6 is the default value).

archive-async

Enables asynchronous archiving of WAL files which allows a higher archiving throughput.

spool-path

Where to keep additional information for asynchronous archiving (status dir).

archive-queue-max

How many WAL segments to keep before throwing segments away. *Note: We configure a value of 1TB to ensure pgbackrest never throws WAL segments away by default*

repo-path

This sets the main directory where backups and WAL files are stored in. It can be set to any desired mount point so backups to remote servers are easily possible.

4.7.5 Backup

For each cluster there is a systemd service which does a full or incremental backup.

- `pgbackrest@<version>-<name>`
- `pgbackrest-incr@<version>-<name>`

To create an ad-hoc backup the corresponding service can be started. `systemctl start pgbackrest@9.6-main` would create a full backup of the cluster 9.6-main.

If no previous full backup is available `pgbackrest-incr@` will also create a full backup.

4.7.6 Automation

To automate the creation of backups and the retention policy enforcing there are two systemd timers per cluster.

- `pgbackrest@-.timer`
- `pgbackrest-incr@-.timer`

`pgbackrest@<version>-<name>.timer` triggers full backups and `pgbackrest-incr@<version>-<name>.timer` triggers incremental backups.

These timers are created for every cluster and are initialized with a default timing. The timers can be enabled independently for every database cluster either via systemd or the web portal. To fully enable a timed backup the timer must be *started* **and** *enabled*. If the timer is *started* but not *enabled* systemd will not start it after the next reboot.

Keep in mind that enabling only the incremental backup is only reasonable for shorter periods of time, special scenarios like not changing databases, or if the full backups are triggered in another way. To keep storage and restore time at an reasonable level periodic full backups are needed.

pgbackrest@.timer

```
# /lib/systemd/system/pgbackrest@.timer
[Unit]
Description=Automated pgBackRest full backup of PostgreSQL cluster %i

[Timer]
OnCalendar=Sun *--* 01:00:00
RandomizedDelaySec=2h

[Install]
WantedBy=multi-user.target
```

This timer triggers a full backup every Sunday in the early morning 01:00 or randomly up to 2 hours later. The random delay set by `RandomizedDelaySec=2h` is set so systemd can schedule many timers over a given time range. Here it is done so that not all backups for all clusters start at the same time blocking the I/O.

pgbackrest-incr@.timer

```
# /lib/systemd/system/pgbackrest-incr@.timer
[Unit]
Description=Automated pgBackRest incremental backup of PostgreSQL cluster %i

[Timer]
OnCalendar=Tue,Thu *--* 01:00:00
RandomizedDelaySec=2h

[Install]
WantedBy=multi-user.target
```

This timer triggers an incremental backup every Tuesday and Thursday in the early morning 01:00 or randomly up to 2 hours later.

WAL Archiving

WAL archiving is disabled by default for new PostgreSQL clusters. It can be activated using the portal (see [portal](#)) or by starting `pgbackrest-toggle-archiving.service`. The service toggles archiving mode to on or off, depending on the former state.

Note: If archiving is disabled and a full or incremental backup is started (manual or via timer), archiving is automatically enabled. This step is required to ensure all WAL files need for a restore are archived beside the basebackup. **Archiving is *not* disabled after the backup run.**

4.7.7 Restore

Restore is an invasive action that can destroy data if not executed properly!

To restore a backup there are two main methods full and delta.

Full Restore

A full restore restores a given backup (by default the latest) to the given (default) destination. The restore command expects the target directory to be empty. This can be used to setup a cluster on a new machine, small clusters or if most of the remaining data is incorrect.

Steps to full restore.

1. Stop the cluster (if still running)
2. Delete or move all remaining data
3. Restore full content from backup

All steps should be run as user postgres.

```
# 1. Stop the cluster
pg_ctlcluster <major version> <name> stop

# 2. Delete or move all remaining data
mv /var/lib/postgresql/<major version>/<name> /var/somewhere-save
mkdir /var/lib/postgresql/<major version>/<name>

# 3. Restore full content from backup
pgbackrest --stanza=<major version>-<name> restore
```

After this the cluster can be started again. If there is enough storage available it should be preferred to move the data to a save place instead of deletion.

Delta Restore

A delta restore does not need a clean target and it only copies files that differ from backup. This approach can be much faster especially if most of the underling files did not change since the last backup.

This has the potential to destroy data! Because this works on the cluster data it is possible to cause damage. Data that is not in the backup / WAL archive but in the current cluster will be lost!

Steps to perform a delta restore.

1. Stop the cluster (if still running)
2. Restore delta content from backup

```
# 1. Stop the cluster
pg_ctlcluster <major version> <name> stop

# 2. Restore full content from backup
pgbackrest --stanza=<major version>-<name> --delta restore
```

After this the cluster can be started again.

Point in Time Recovery

The shown backups methods do a full restore. This means a all basebackup files and copied back from the archive and all WAL files are applied.

If another recovery target should be restored `--type` and `--target` must be specified. Most of the time one would like to restore a database to a given point in time (e.g. '2017-08-24 12:00:00'). This would require the switch `--type=time` and `--target='2017-08-24 12:00:00'`.

```
pgbackrest --stanza=<major version>-<name> --type=time --target="<ISO timestamp>"
↪ restore
```

4.7.8 Additional Resources

- <http://www.pgbackrest.org/user-guide.html>
- <http://www.pgbackrest.org/command.html>
- <http://www.pgbackrest.org/configuration.html>

4.8 Reporting - pgBadger

A pgBadger service is created for each PostgreSQL instance. Those services are autogenerated and updated each time a new cluster is created or dropped (systemd-generators).

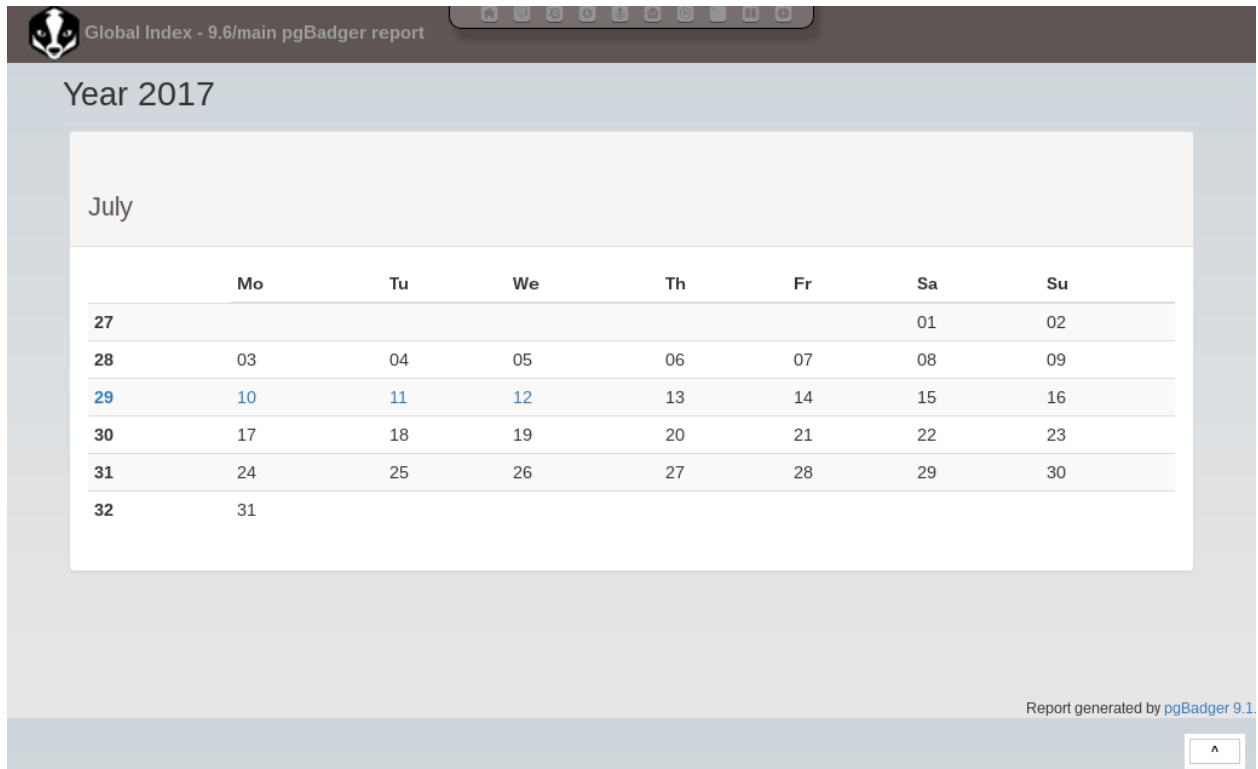
A pgBadger systemd timer ensures reports are updated on a regular basis. By default this is every day at 23:00.

Each pgBadger service parses the PostgreSQL log file of the corresponding PostgreSQL instance. Generated reports are saved within `/var/lib/pgbadger/<version>-<name>` (e.g. `/var/lib/pgbadger/9.6-main/`).

All reports are accessible in the web interface. An calendar provides access to daily and weekly reports.

A manual update of those reports can be triggered either using the corresponding service (e.g. `pgbadger@9.6-main.service`) or using the *portal*. A update of *all* reports could be triggered using the parent service `pgbadger.service`.

Note: Changing `postgresql.conf` settings like `log_line_prefix` or `lc_messages` can lead to pgBadger reports not getting updated anymore.



4.9 Web Terminal - Shell In A Box

Shell In A Box is a convenient web based terminal. It can be used like a normal console connection. Explicit login and authentication is required. To change settings (e.g. the color theme) just right click anywhere on the terminal window.

4.9.1 Additional Resources

- <https://github.com/shellinabox/shellinabox>

4.10 Firewall - ferm

By default ferm is installed as a front-end for creating iptables rules. A default config is deployed in the following location:

- `/etc/ferm.conf`
- `/etc/ferm.d/elephant-shed.conf`

The default policy for incoming traffic is set to DROP. Local traffic is allowed and we explicitly whitelisted the following incoming traffic:

- state RELATED,ESTABLISHED
- ICMP (ping)
- SSH
- HTTP / HTTPS
- PostgreSQL: Ports 5432-5439 (first 8 clusters)

Every additional service that should be reachable from the network needs to be whitelisted as well. To configure own firewall rules simply create a ferm configuration within `/etc/ferm.d/` with the extension `.conf`. Those configurations files are loaded by ferm automatically on restart.

Note: Services like Prometheus and Grafana are by default only available via the reverse proxy which manages the authentication. If these services are made reachable over the network, precautions must be taken.

4.10.1 Additional Resources

- <http://ferm.foo-projects.org/>
- <https://wiki.debian.org/ferm>

4.11 Remote Control - tmate

tmate is a fork of the popular terminal multiplexer tmux. It is used to provide remote support if needed.

It is preconfigured to connect to a relay server (`tmate.credativ.com`) and enables the user to share the current terminal with a third party by sending an SSH command including a secret token.

There are two modes of operation, read-write and read-only. This enables the user to give a third party temporary access to the current terminal. The user can always watch the terminal and audit the actions taken by the third party.

- tmate is not running by default, it needs to be started explicitly
- When the initiating shell is closed, the connection is closed as well
- The backend to use is fully configurable (in `/etc/tmate.conf`) and preset to `tmate.credativ.com`
- tmate is included as a technical preview to evaluate the potential

4.11.1 Usage

Start tmate (opens a new terminal)

```
tmate
```

Show the credentials which need to be given to a third party (securely)

```
tmate show-messages
```

```

sosna@ohm ~ % tmate show-messages
Wed Jul 19 13:48:02 2017 [tmate] Connecting to tmate.credativ.com...
Wed Jul 19 13:48:02 2017 [tmate] Note: clear your terminal before sharing readonly access
Wed Jul 19 13:48:02 2017 [tmate] ssh session read only: ssh -p10022 ro-IZDX3ITepLQmREjrbd2QjNjmv@tmate.credativ.com
Wed Jul 19 13:48:02 2017 [tmate] ssh session: ssh -p10022 Cjr4liAzBbfkuMNwbPALMBxyg@tmate.credativ.com
sosna@ohm ~ %

1 [ 0.0%] 4 [ 0.7%] 7 [ 0.0%] 10 [ 0.7%]
2 [ 0.0%] 5 [ 0.0%] 8 [ 1.3%] 11 [ 1.3%]
3 [ 0.0%] 6 [ 0.0%] 9 [ 1.3%] 12 [ 1.3%]
Mem[|||||]8.82G/31.4G Tasks: 191, 932 thr: 1 running
Swp[ 0K/31.9G] Load average: 0.12 0.24 0.45
Uptime: 2 days, 04:32:29

postgres=#

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
7625 sosna 20 0 2751M 423M 377M S 5.9 1.3 1h03:14 /usr/lib
7636 sosna 20 0 2751M 423M 377M S 2.0 1.3 7:36.97 /usr/lib
7639 sosna 20 0 2751M 423M 377M S 1.3 1.3 29:13.50 /usr/lib
22381 sosna 20 0 25920 4932 3044 S 1.3 0.0 0:03.87 htop
22922 sosna 20 0 26160 5364 2944 R 0.7 0.0 0:01.21 htop
2652 sosna 20 0 3590M 826M 216M S 0.7 2.6 22:02.82 /usr/lib
7637 sosna 20 0 2751M 423M 377M S 0.7 1.3 5:50.99 /usr/lib
2789 sosna 20 0 684M 93756 33900 S 0.7 0.3 2:24.50 /usr/bin
2129 sosna 20 0 2709M 448M 87028 S 0.0 1.4 1h46:33 /usr/bin
2132 sosna 20 0 2709M 448M 87028 S 0.0 1.4 23:37.19 /usr/bin
2013 sosna 20 0 1647M 237M 182M S 0.0 0.7 44:45.82 /usr/lib
2712 sosna 20 0 3590M 826M 216M S 0.0 2.6 1:11.91 /usr/lib
1613 root 20 0 1208M 8984 7376 S 0.0 0.0 0:49.54 docker-c
2023 sosna 20 0 1647M 237M 182M S 0.0 0.7 0:56.88 /usr/lib
2718 sosna 20 0 1887M 668M 154M S 0.0 2.1 50:50.64 /usr/lib
7635 sosna 20 0 2751M 423M 377M S 0.0 1.3 5:48.39 /usr/lib
2724 sosna 20 0 3590M 826M 216M S 0.0 2.6 0:20.98 /usr/lib
2802 sosna 20 0 975M 273M 92236 S 0.0 0.8 1h27:47 /usr/lib
7634 sosna 20 0 2751M 423M 377M S 0.0 1.3 5:50.45 /usr/lib
7638 sosna 20 0 2751M 423M 377M S 0.0 1.3 6:44.23 /usr/lib
1081 Debian-gd 20 0 429M 58996 41228 S 0.0 0.2 0:12.95 /usr/lib
31328 sosna 20 0 1052M 129M 64064 S 0.0 0.4 0:06.08 /usr/lib
2756 sosna 20 0 1887M 668M 154M S 0.0 2.1 14:53.00 /usr/lib
2807 sosna 20 0 975M 273M 92236 S 0.0 0.8 22:04.60 /usr/lib
2014 sosna 20 0 1647M 237M 182M S 0.0 0.7 8:27.00 /usr/lib
29609 sosna 20 0 886M 98776 48000 S 0.0 0.3 0:22.55 pidgin
F1Help F2Setup F3SearchF4FilterF5Free F6SortByF7Nice F8Nice F9Kill
tmate.credativ.com ohm 0:zsh* "ohm" 13:49 19-Jul-17

```

For further usage see the following additional resources regarding tmux.

4.11.2 Additional Resources

- <https://tmate.io/>
- <https://man.openbsd.org/OpenBSD-current/man1/tmux.1>

4.12 Configuration Revision - etckeeper

etckeeper is a set of tools and hooks to keep all configuration in `/etc` in a git repository. Commits can be done manually or will happen automatically via time or by package manager hooks.

Configuration changes can be seen and compared to previous versions. If necessary previous settings can be restored.

4.12.1 Additional Resources

- <https://etckeeper.branchable.com/>
- https://www.thomas-krenn.com/de/wiki/Etc-Verzeichnis_mit_etckeeper_versionieren

CHAPTER 5

Users

The web interface is password protected (HTTP basic authentication) and uses the system users via PAM. When deployed via Ansible, the initial user is **admin** with password **admin**. This user works for web access as well as for SSH and PostgreSQL.

To create new users, use `adduser`, and add the user to the **elephant-shed** group.

```
adduser myon  
adduser myon elephant-shed
```


The following PostgreSQL versions are currently supported:

- 10
- 9.6
- 9.5
- 9.4

Other versions are usable but not fully integrated in all components. This might make additional manual tuning necessary.

All official supported versions are installable via the configured *apt.postgresql.org* repository. Please see <https://apt.postgresql.org/> for more information. We intend to support newer versions of PostgreSQL as soon as they are released.

Known Bugs and Issues

7.1 PostgreSQL

- The `prometheus-sql-exporter` monitoring agent is permanently keeping connections open to all databases, which prevents `DROP DATABASE` from working. To drop databases, stop `prometheus-sql-exporter` first. This is possible via the web interface Cockpit: [services#/prometheus-sql-exporter.service](#).

7.2 pgadmin4

- `pgadmin4` does not use PAM authentication.
- By default, `pgadmin4` does not show local databases.

7.3 Portal

- A direct relogin after a logout does not work. Reloading the page is necessary.

7.4 Cockpit

- The current version of `cockpit-packagekit` is not usable on `ppc64el` systems due to a bug.

CHAPTER 8

License

The Elephant Shed itself is licensed under the GPLv3 (<https://www.gnu.org/licenses/gpl-3.0.de.html>).

All bundled components are Free/Open-Source software with a known and approved open source license.

9.1 Do you have any question or want to know more?

- **Project page** elephant-shed.io
- **Git** github.com/credativ/elephant-shed
- **Web-Chat** [#elephant-shed](#)
- **IRC** [#elephant-shed](#) on [irc.oftc.net](#)

9.2 Do you need professional support or additional services?

Elephant Shed is an open source project, developed and maintained by credativ.

For the Elephant Shed PostgreSQL appliance, credativ offers comprehensive technical support with service level agreements, which are also available on 365 days a year and 24 hours a day as an option.

Installation and integration support, as well as an introduction in Elephant Shed PostgreSQL appliance is of course also part of credativ's services. If you are interested, please feel free to contact us.



- **Web** credativ.de
- **E-Mail:** info@credativ.de
- **Phone:** +49 2166 9901-0