WSP

Web Services Platform

A friendly control panel for clouds that takes care of DevOps while you focus on developing your app

plesk

Goal: **host my web application with failover, scaling and CI/CD**

Let’s use

Clouds (IaaS)

AWS, Google Cloud, Azure, etc

Problem: **Complexity**

How to solve the problem:
experienced DevOps engineers to:

- **analyse** the web app
- **design** the architecture
- **configure** the cloud components
- **set up** auxiliary services like logging, monitoring and alerting
- **maintain** all the stuff doing regular analysis and improvements

Problems with DevOps engineers:

1. human resource isn’t scalable
2. costs (experienced DevOps are expensive)

Goal: **host my web application with failover, scaling and CI/CD**

Let’s use

PaaS

Heroku, Pantheon, etc

Problems:

- **inconvenient pricing plans,** long plan switching is the problem in case of the slashdot effect; you don’t know what exactly you are paying for
- **no lift & shift:** most PaaSes force you to change you app to fit into them
- **vendor lock:** once you’ve started using the PaaS, you can't easily stop using it or migrate to another solution
- **limited customisations, no control,** e.g. a limited set of available PHP modules

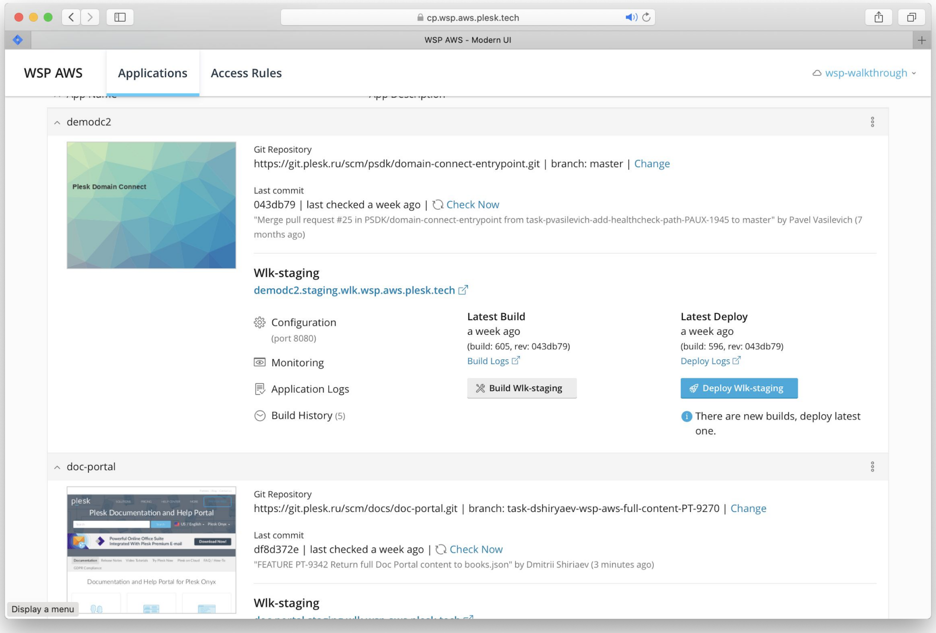
How to solve the problems: no answer

Goal: **host my web application with failover, scaling and CI/CD**

WSP

Web Services Platform

Easy to use - DevOps not required
Your app is on the air within 15 minutes
Manages your AWS account
Integrates with AWS Cost Explorer
Lift & Shift
No vendor lock
Full control and freedom
Autoconfigured CI/CD
Made by Plesk


cp.wsp.aws.plesk.tech

WSP AWS - Modern UI

WSP AWS | Applications | Access Rules | wsp-walkthrough

demodc2

Git Repository
https://git.plesk.ru/scm/psdk/domain-connect-entrypoint.git | branch: master | Change

Last commit
043db79 | last checked a week ago | Check Now
"Merge pull request #25 in PSDK/domain-connect-entrypoint from task-pvasilevich-add-healthcheck-path-PAUX-1945 to master" by Pavel Vasilevich (7 months ago)

Wlk-staging
demodc2.staging.wlk.wsp.aws.plesk.tech

Configuration (port 8080)
Monitoring
Application Logs
Build History (5)

Latest Build
a week ago
(build: 605, rev: 043db79)
Build Logs
Build Wlk-staging

Latest Deploy
a week ago
(build: 596, rev: 043db79)
Deploy Logs
Deploy Wlk-staging
There are new builds, deploy latest one.

doc-portal

Git Repository
https://git.plesk.ru/scm/docs/doc-portal.git | branch: task-dshiryaev-wsp-aws-full-content-PT-9270 | Change

Last commit
df8d372e | last checked a week ago | Check Now
"FEATURE PT-9342 Return full Doc Portal content to books.json" by Dmitrii Shiriaev (3 minutes ago)

Wlk-staging

Plesk Domain Connect
Plesk Documentation and Help Portal
Documentation and Help Portal for Plesk Onyx
Display a menu

How does it look **now**?

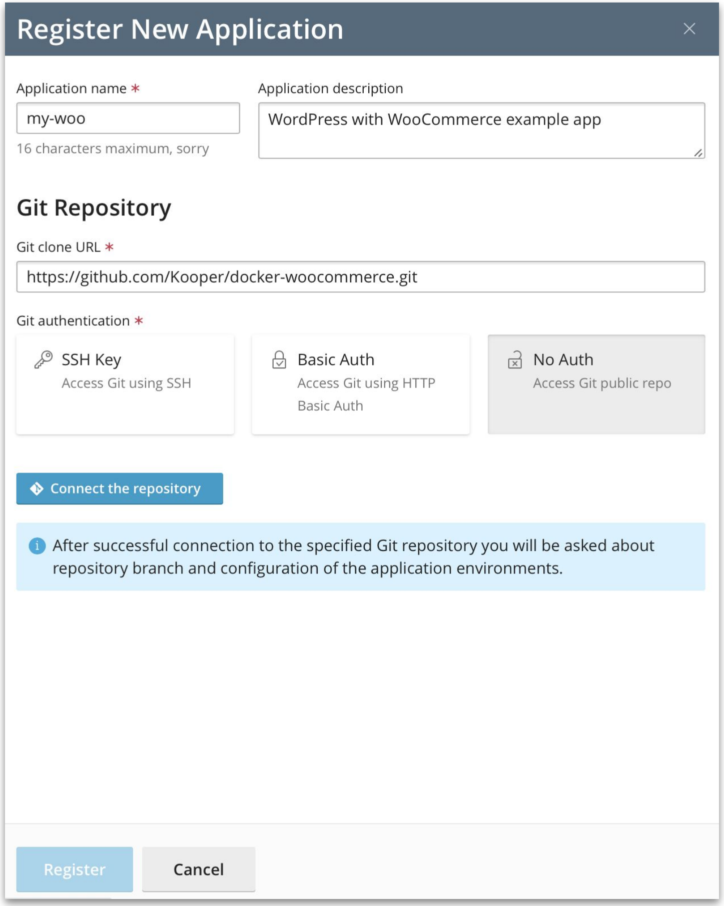
Step 1 **Connect** your AWS account to WSP
using the provided CloudFormation script

Step 2 **Register** your web app in WSP
specifying the git clone URL and other app parameters

Step 3 **Build** and **deploy** your app into your AWS account
by clicking the “Build” and “Deploy” buttons in the UI

Step 4 **Commit** changes, rebuild and redeploy your app
automatically or by clicking the buttons in the UI

Screenshots



Register New Application

Application name * my-woo
16 characters maximum, sorry

Application description: WordPress with WooCommerce example app

Git Repository

Git clone URL * https://github.com/Kooper/docker-woocommerce.git

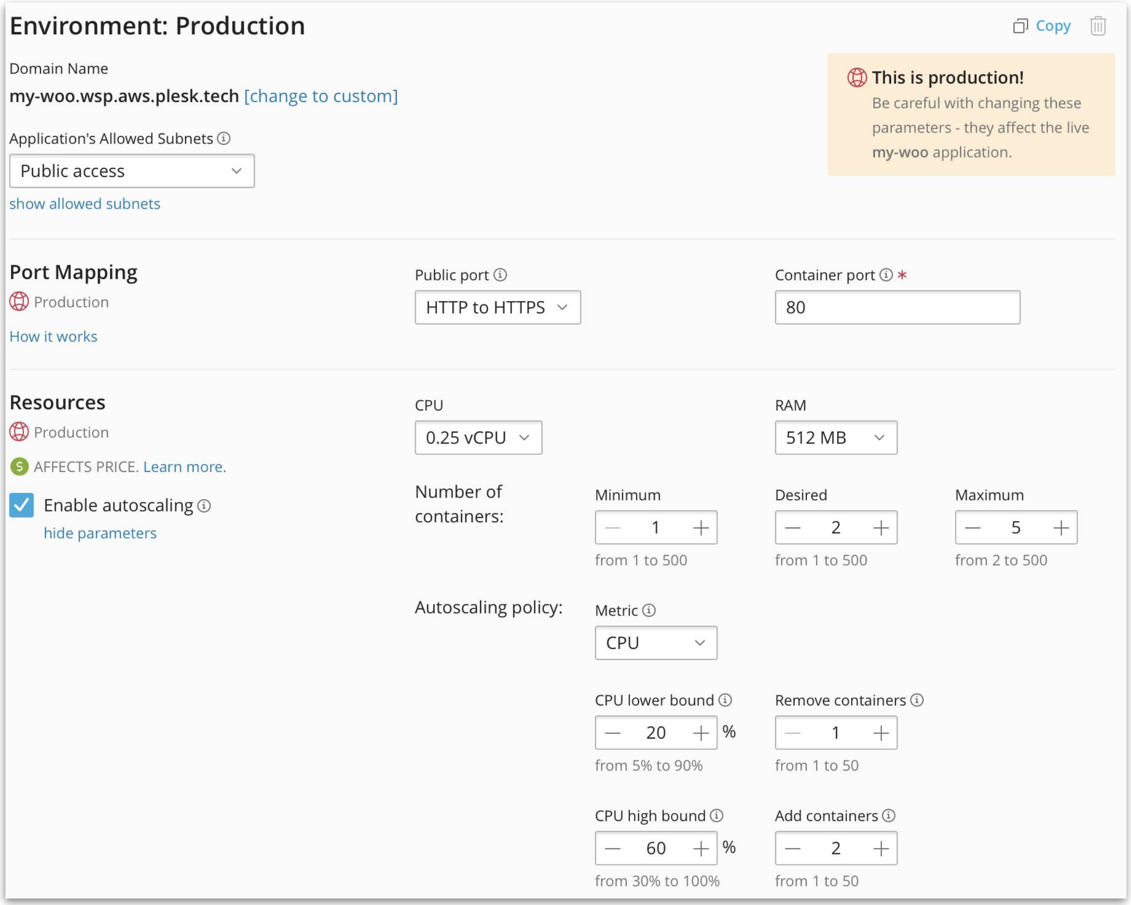
Git authentication *

- SSH Key — Access Git using SSH
- Basic Auth — Access Git using HTTP Basic Auth
- [✓ No Auth — Access Git public repo]

Connect the repository

After successful connection to the specified Git repository you will be asked about repository branch and configuration of the application environments.

Register | Cancel



Environment: Production

Copy

Domain Name
my-woo.wsp.aws.plesk.tech [change to custom]

Application's Allowed Subnets: Public access
show allowed subnets

This is production!
Be careful with changing these parameters - they affect the live **my-woo** application.

Port Mapping
Production
How it works

Public port: HTTP to HTTPS
Container port *: 80

Resources
Production
AFFECTS PRICE. Learn more.
✓ Enable autoscaling
hide parameters

CPU: 0.25 vCPU
RAM: 512 MB

Number of containers:

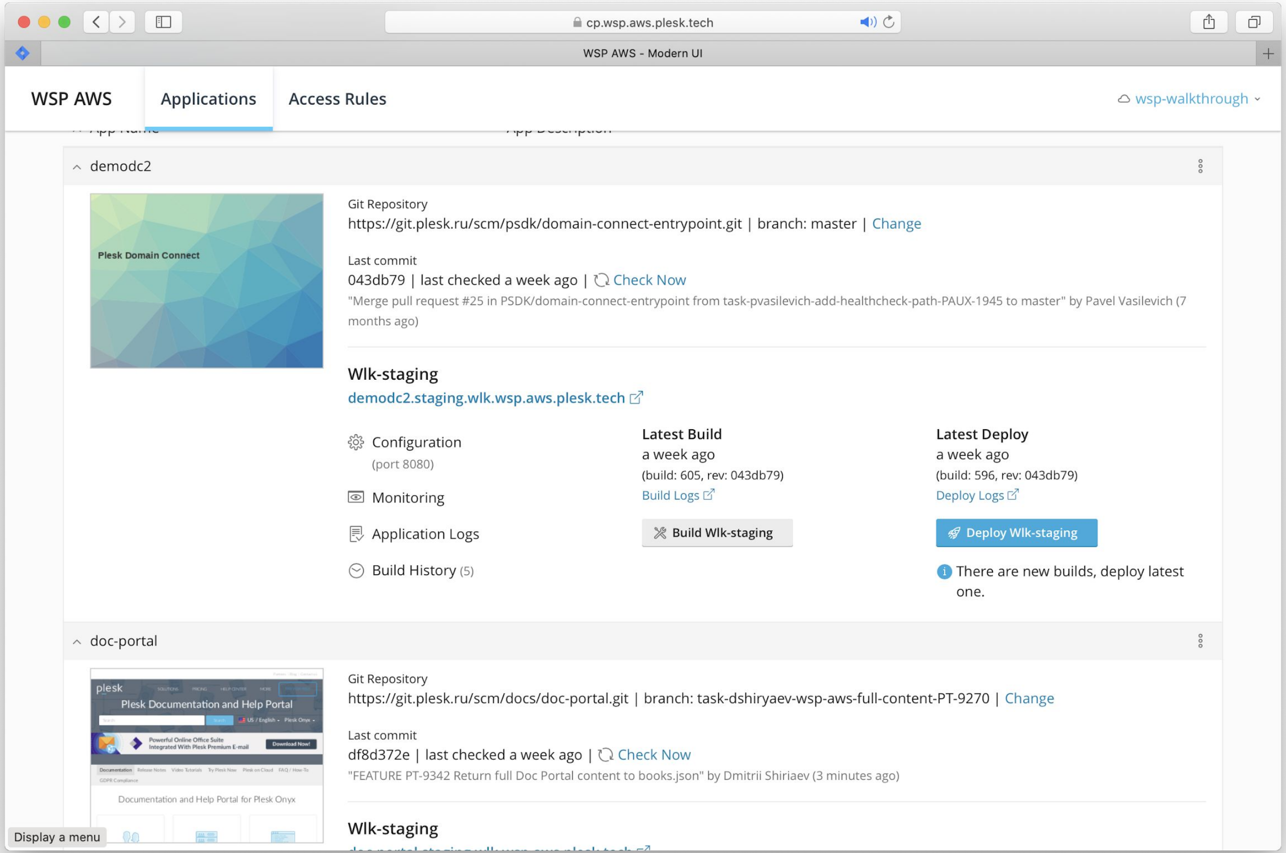
Minimum	Desired	Maximum
1	2	5
from 1 to 500	from 1 to 500	from 2 to 500

Autoscaling policy:

Metric: CPU

CPU lower bound: 20 %	Remove containers: 1
from 5% to 90%	from 1 to 50
CPU high bound: 60 %	Add containers: 2
from 30% to 100%	from 1 to 50

Screenshots



demodc2

Git Repository
https://git.plesk.ru/scm/psdk/domain-connect-entrypoint.git | branch: master | Change

Last commit
043db79 | last checked a week ago | Check Now
"Merge pull request #25 in PSDK/domain-connect-entrypoint from task-pvasilevich-add-healthcheck-path-PAUX-1945 to master" by Pavel Vasilevich (7 months ago)

Wlk-staging
demodc2.staging.wlk.wsp.aws.plesk.tech

Configuration (port 8080)
Monitoring
Application Logs
Build History (5)

Latest Build
a week ago
(build: 605, rev: 043db79)
Build Logs
Build Wlk-staging

Latest Deploy
a week ago
(build: 596, rev: 043db79)
Deploy Logs
Deploy Wlk-staging
There are new builds, deploy latest one.

doc-portal

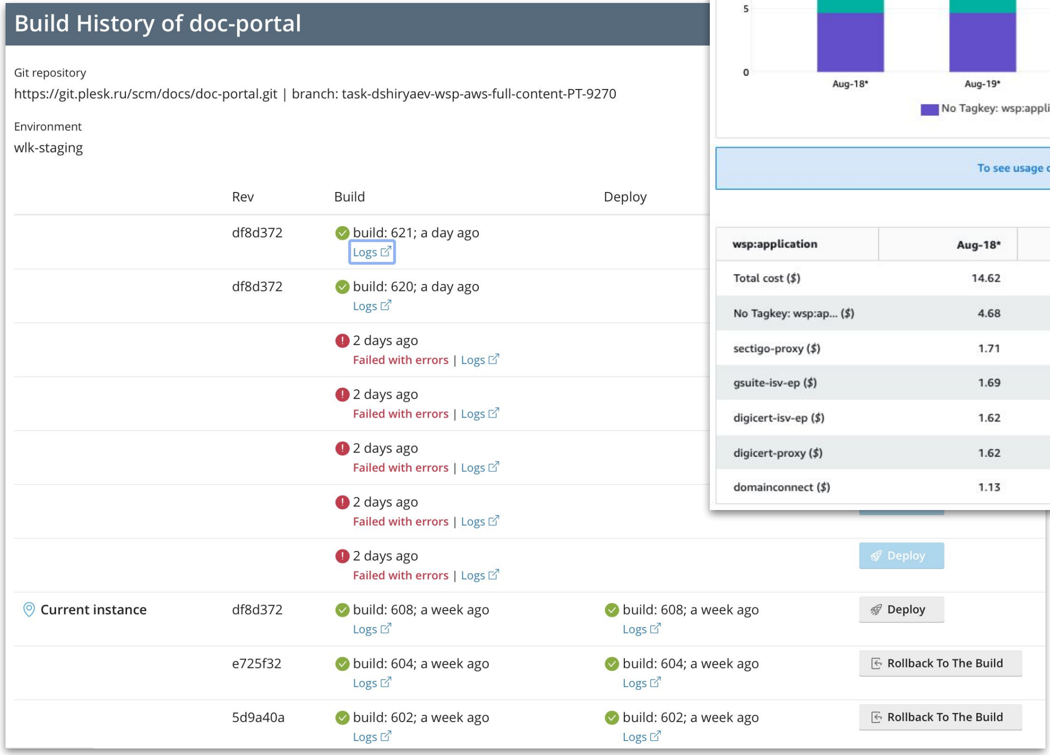
Git Repository
https://git.plesk.ru/scm/docs/doc-portal.git | branch: task-dshiryaev-wsp-aws-full-content-PT-9270 | Change

Last commit
df8d372e | last checked a week ago | Check Now
"FEATURE PT-9342 Return full Doc Portal content to books.json" by Dmitrii Shiriaev (3 minutes ago)

Wlk-staging

Screenshots

Build History

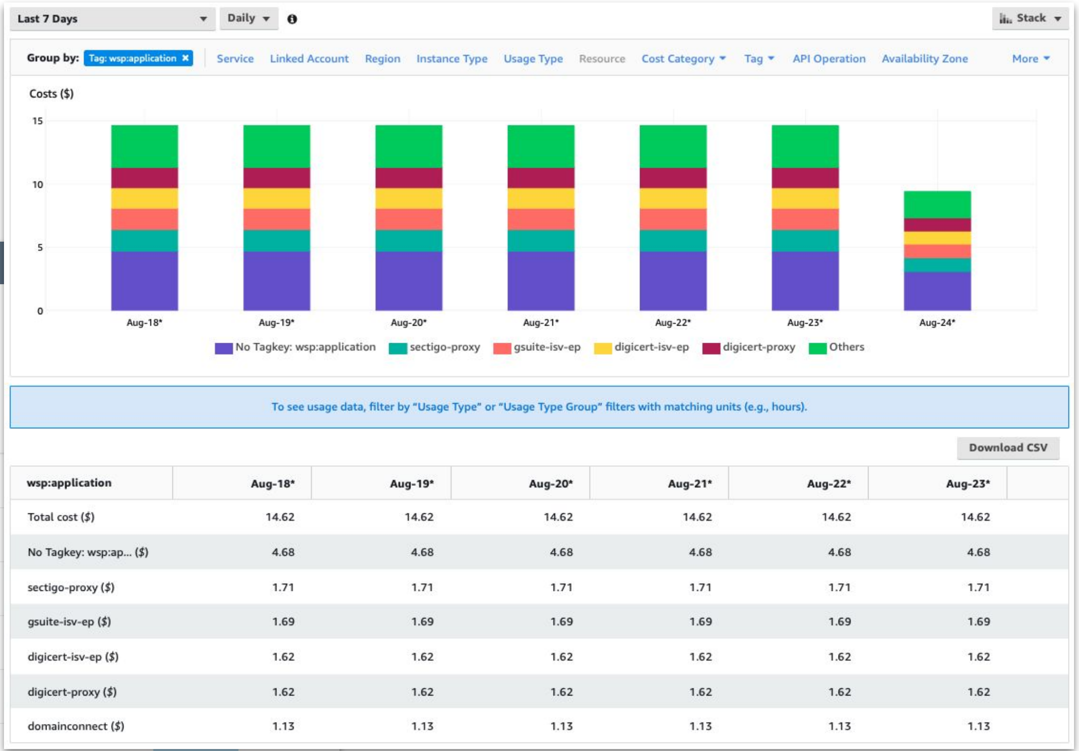


Build History of doc-portal

Git repository
https://git.plesk.ru/scm/docs/doc-portal.git | branch: task-dshiryaev-wsp-aws-full-content-PT-9270

Environment
wlk-staging

	Rev	Build	Deploy	
	df8d372	✓ build: 621; a day ago Logs		
	df8d372	✓ build: 620; a day ago Logs		
		! 2 days ago Failed with errors | Logs		
		! 2 days ago Failed with errors | Logs		
		! 2 days ago Failed with errors | Logs		
		! 2 days ago Failed with errors | Logs		
		! 2 days ago Failed with errors | Logs		Deploy
Current instance	df8d372	✓ build: 608; a week ago Logs	✓ build: 608; a week ago Logs	Deploy
	e725f32	✓ build: 604; a week ago Logs	✓ build: 604; a week ago Logs	Rollback To The Build
	5d9a40a	✓ build: 602; a week ago Logs	✓ build: 602; a week ago Logs	Rollback To The Build



To see usage data, filter by "Usage Type" or "Usage Type Group" filters with matching units (e.g., hours).

wsp:application	Aug-18*	Aug-19*	Aug-20*	Aug-21*	Aug-22*	Aug-23*
Total cost ($)	14.62	14.62	14.62	14.62	14.62	14.62
No Tagkey: wsp:ap... ($)	4.68	4.68	4.68	4.68	4.68	4.68
sectigo-proxy ($)	1.71	1.71	1.71	1.71	1.71	1.71
gsuite-isv-ep ($)	1.69	1.69	1.69	1.69	1.69	1.69
digicert-isv-ep ($)	1.62	1.62	1.62	1.62	1.62	1.62
digicert-proxy ($)	1.62	1.62	1.62	1.62	1.62	1.62
domainconnect ($)	1.13	1.13	1.13	1.13	1.13	1.13

Cost Explorer*

*if you've enabled customer Cost allocation tags (manual action because AWS has no API for that)

Check if you can use WSP

the app runs as a **Docker** container

and can be built with the `docker build` command

you own an **AWS** account

- future or a **k8s** cluster
- future or **nothing** if you're OK with using our AWS resources
- far future or an account in **GCP**, **Azure**, other clouds

git repo of your app is accessible world-wide

not to worry, we're talking about network. The repo itself can be private.

- future or WSP can use a **docker registry** (**DockerHub**)
- future or you can provide WSP with the image as a **file**

WSP highlights: **principles**

- **SaaS** no need to install and maintain it
- **friendly UX** you don't have to be an experienced AWS engineer
- **lift&shift** no need to modify your application to run it with WSP
- **security** secrets (passwords) are encrypted, also, WSP automatically configures VPCs to provide network isolation
- **no vendor lock** If you stop using WSP, your apps will still be configured and working. WSP just configures services in your AWS account.

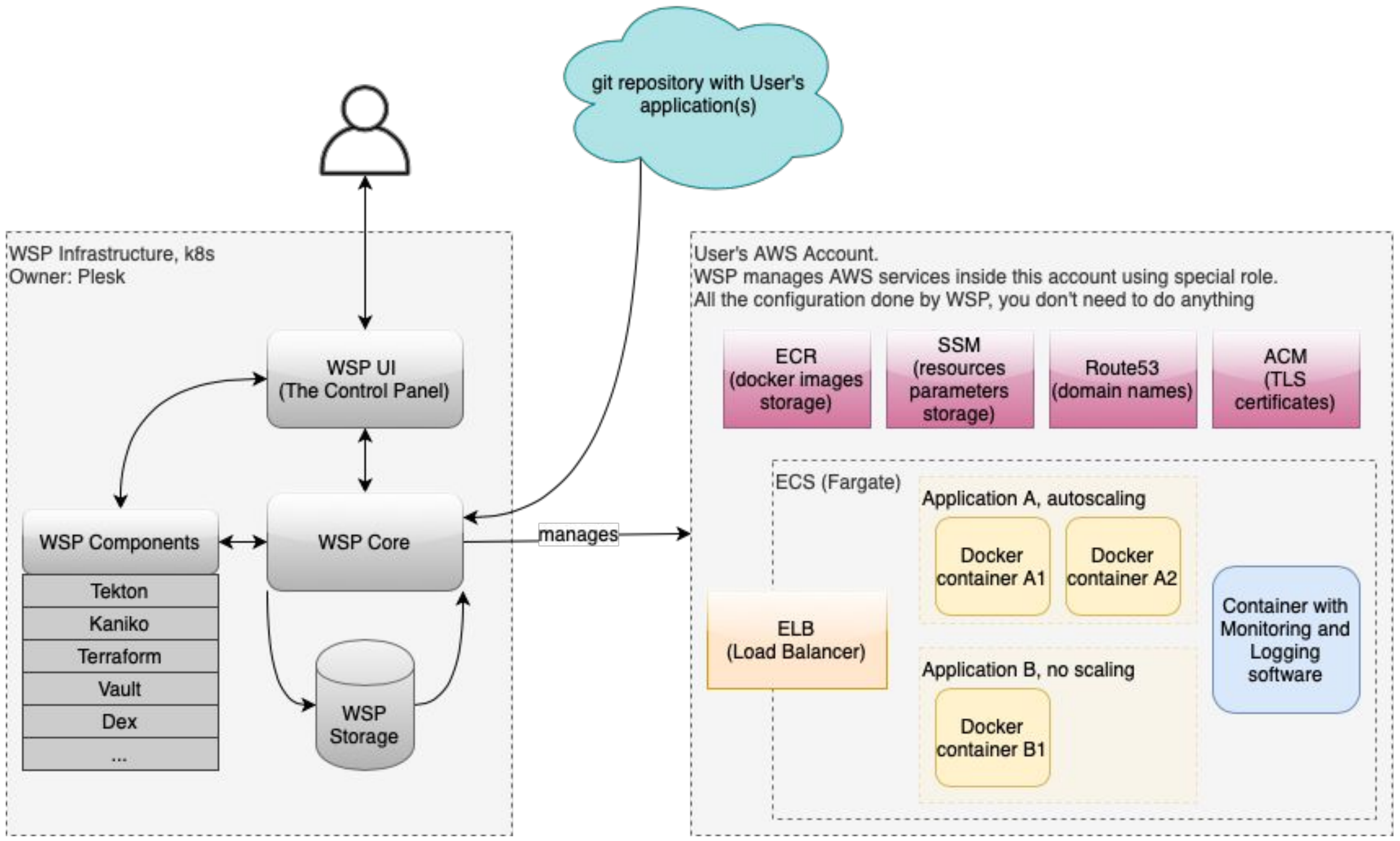
WSP highlights: **features**

- manual vertical scaling, **automated horizontal scaling**.
- **monitoring** and **logging** are set up and configured automatically for every application. It's hard to set up monitoring and logging properly. Especially in AWS. Especially cheap. WSP does it for you.
    - future **real-time logging**. Usually, logging in AWS is buffered a lot, so, you don't see what's going on right now.
    - future **grafana** dashboard presets
    - future **alerting** presets
- **builds history** - you can easily roll back to previous builds
- **environments** - deploy your app in staging, production, or production in a particular region
- DevOps practices
    - **infrastructure as a code**: if the git repo contains the WSP manifest, WSP will read it
        - future WSP also updates that manifest according to actions in the UI
    - future full **GitOps** support
        - already available **build and deploy on commit**

WSP highlights: **features**

- technical **domain names** configured automatically. Technical domains are **secured with free TLS certificates**. Custom domain names are also supported.
- management of such AWS services as **SES** and **RDS**.
    - **future** other AWS services
- WSP sets special tags when configures your apps, so you can see **detailed costs** in **AWS Costs Explorer**. Without such tags, you can only see the total costs without details.
- **zero downtime** when you deploy a new version of your app - load balancer routes traffic to the old version while the new version is starting.
- **access restrictions** management - allow access to you app for particular IPs only.
    - **future** **secret links** for temporary access (e.g., demo the staging instance of the app to your customer)
- **future** **team management** - roles and permissions
- **asynchronous UI**
- WSP built on top of **industry standard components**

Principal scheme



If you’d like to try the solution, contact us.

We also welcome your feedback!