Drupal Camp

Berlin 2024

Key Technical Decisions framework

Engin Yilmaz

- Solutions Architect at Jakala
- Since 2015 in the organization
- Bachelor of Science in Informatik
- ❤️ Open Source
- 2 daughters 👧👧

GLOBAL SCALE, LOCAL PRESENCE - WORKING AS ONE

Revenues
500M +

Growth rate
+19%
2018-2023 CAGR

Offices in
20+
Countries

Clients
900

Active
Projects in
30+
Countries

We are also present in: Gabrovo - Veliko Tarnovo - Plovdiv - Gijon – Turin – Macerata – Naples – Bari – Rende – Catania – Vienna - Luxembourg – Düsseldorf – Riga

Summary

- Introduction - what is this about?
- What is a KTD?
- Anatomy of a KTD
- Why KTDs?
- Other standards

What is this about?

The average digital project has changed significantly.

From sizable web builds we are now landing contracts for multi-million dollar composable digital experience platforms.

With decoupled front-ends, elaborate system architectures and multiple third party integrations.

Infrastructure needs have changed too.

From a few servers with simple go to market mechanisms.

To containerised public clouds, distributed file systems and high availability databases, with complex integration and deployment pipelines.

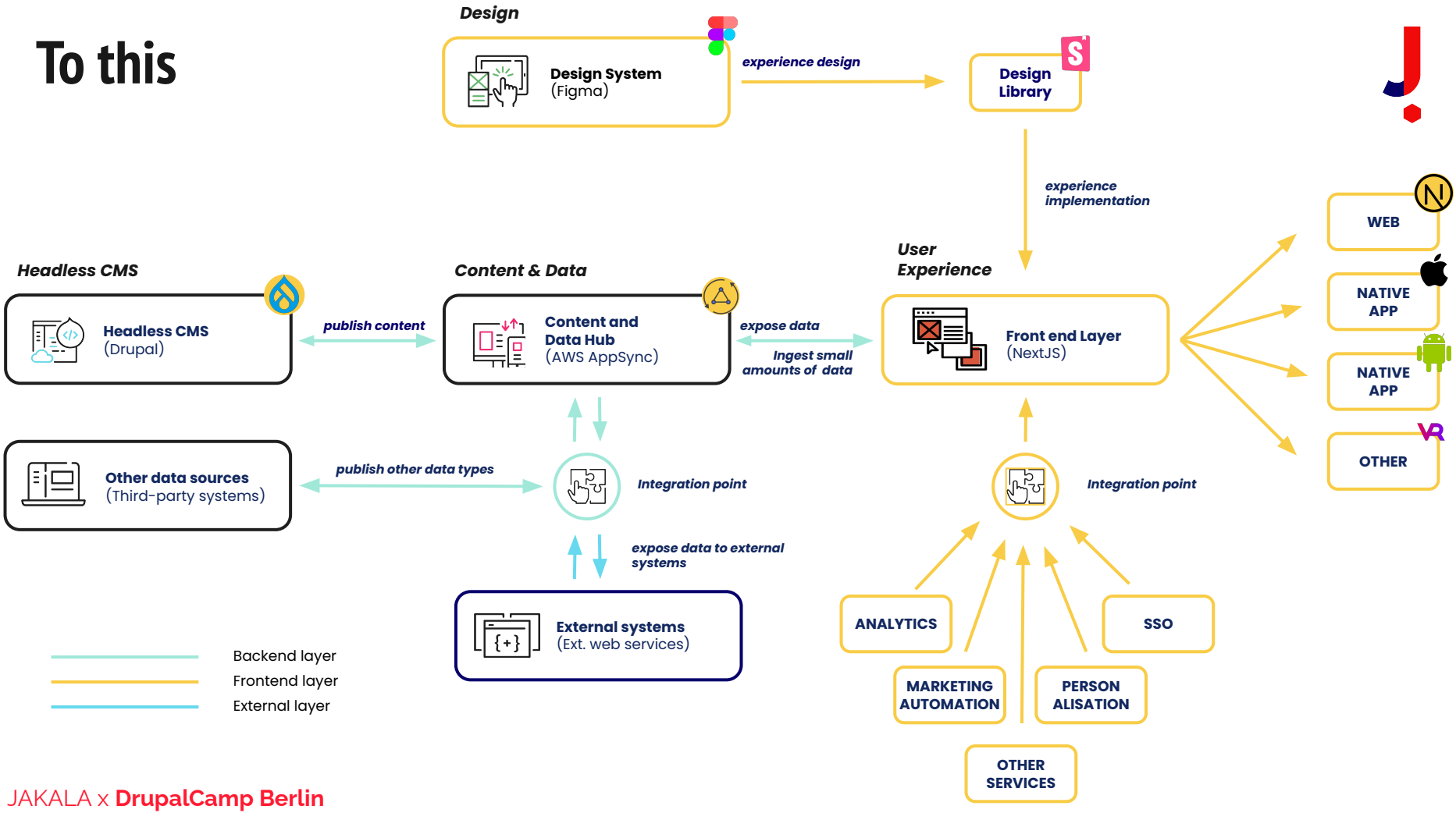
Our ways of working have shifted also.

- From a couple of developers and few photoshop files, finished before development started.

- To multidisciplinary agile teams
- iterative component based design
- decoupled JS front-end frameworks
- third-party component libraries
- and multiple touchpoints and channels.

We moved from this

To this


Design
Design System
(Figma)
experience design
Design
Library
experience
implementation
Headless CMS
Headless CMS
(Drupal)
publish content
Content & Data
Content and
Data Hub
(AWS AppSync)
expose data
Ingest small
amounts of data
User
Experience
Front end Layer
(NextJS)
WEB
NATIVE
APP
NATIVE
APP
OTHER
Other data sources
(Third-party systems)
publish other data types
Integration point
expose data to external
systems
External systems
(Ext. web services)
Integration point
ANALYTICS
SSO
MARKETING
AUTOMATION
PERSON
ALISATION
OTHER
SERVICES
Backend layer
Frontend layer
External layer

Digital does <u>not</u> have an end date.

- Today everything is in **product-mode**.
- Our work often is not governed by a beginning, a middle and an end.

- We are working on products, platforms or services, with a **fluid** scope, **flexible** timelines and **adjustable** budgets.
- Our work often needs refinement, refactoring, adaptation.

All of that comes with a whole set of challenges.

How do you make the **best** technical decisions in such a **dynamic** environment?

How do you know if an approach is good enough for now and effectively manage scope?

How do you quantify and manage the technical debt introduced?

How do you **document** those decisions so they can be **revisited** later on?

How do you give **voice** to everyone who has something valuable to contribute?

How do you create **consensus** within your team?

What is a KTD?

Key technical Decisions

Are decisions, that have profound strategic and architectural implications, for the organisation.

They enable or inhibit strategic goals: evolution, growth and value creation.

KTDs are not User Stories

User stories are about technical implementation details, functionality.

They still may require discussion over details, but the approach has already been given.

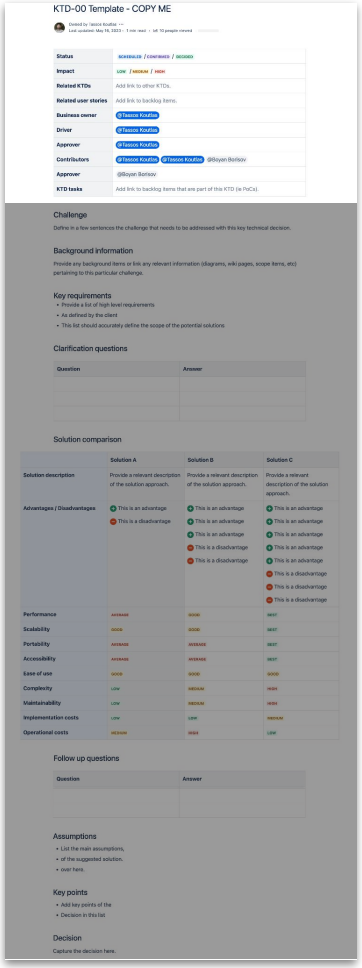
User Stories influence project delivery, not strategy.

The KTD process

The KTD process is a lightweight but powerful methodology for driving key technical decisions when implementing digital products. Organisations are able to accommodate change in dynamic environments and create alignment, consensus and trust with stakeholders.

The KTD process can work alongside waterfall or agile methods. For agile methods, the sprint cadence needs to be combined with KTD process so one informs the other during project delivery. For waterfall the workflow is straightforward, make all the decisions prior to technical project delivery.

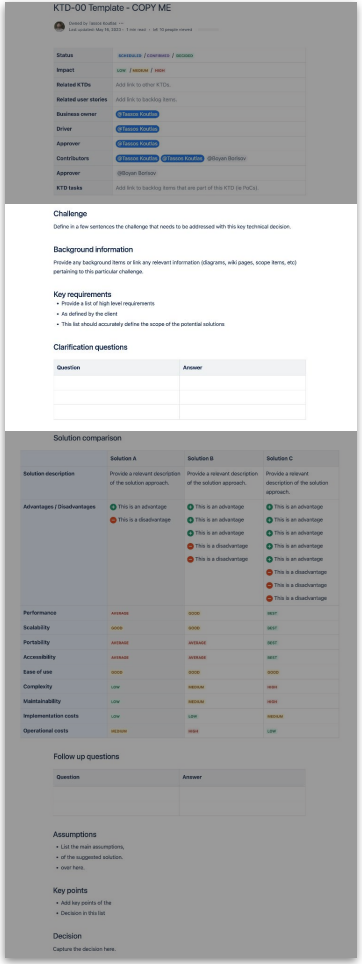
Anatomy of a KTD



KTD-00 Template - COPY ME

Owned by Tassos Koutlas •••
Last updated: May 16, 2023 • 1 min read • 10 people viewed

Status	SCHEDULED / CONFIRMED / DECIDED
Impact	LOW / MEDIUM / HIGH
Related KTDs	Add link to other KTDs.
Related user stories	Add link to backlog items.
Business owner	@Tassos Koutlas
Driver	@Tassos Koutlas
Approver	@Tassos Koutlas
Contributors	@Tassos Koutlas @Tassos Koutlas @Boyan Borisov
Approver	@Boyan Borisov
KTD tasks	Add link to backlog items that are part of this KTD (ie PoCs).



Challenge

Define in a few sentences the challenge that needs to be addressed with this key technical decision.

Background information

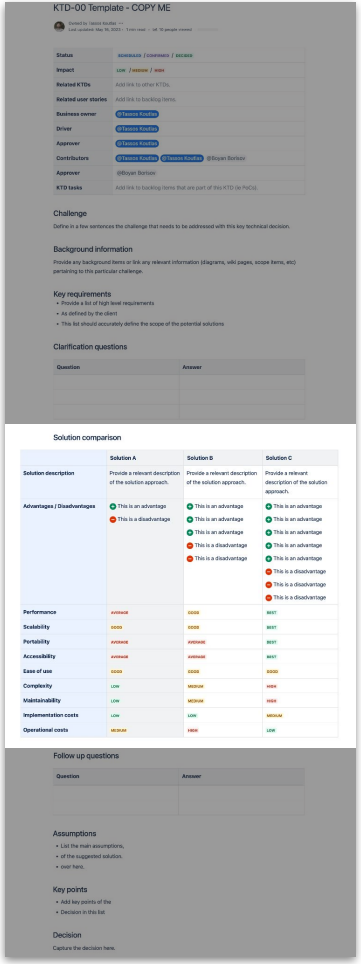
Provide any background items or link any relevant information (diagrams, wiki pages, scope items, etc) pertaining to this particular challenge.

Key requirements

- Provide a list of high level requirements
- As defined by the client
- This list should accurately define the scope of the potential solutions

Clarification questions

Question	Answer



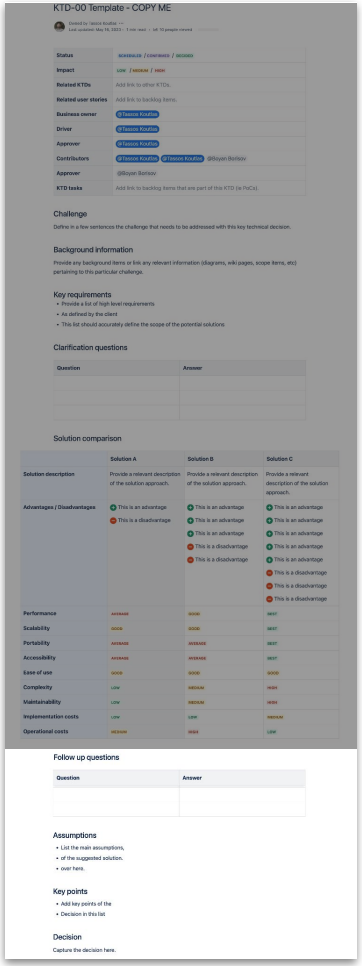
Solution comparison

	Solution A	Solution B	Solution C
Solution description	Provide a relevant description of the solution approach.	Provide a relevant description of the solution approach.	Provide a relevant description of the solution approach.
Advantages / Disadvantages	⊕ This is an advantage ⊖ This is a disadvantage	⊕ This is an advantage ⊕ This is an advantage ⊕ This is an advantage ⊖ This is a disadvantage ⊖ This is a disadvantage	⊕ This is an advantage ⊕ This is an advantage ⊕ This is an advantage ⊕ This is an advantage ⊕ This is an advantage ⊖ This is a disadvantage ⊖ This is a disadvantage ⊖ This is a disadvantage
Performance	AVERAGE	GOOD	BEST
Scalability	GOOD	GOOD	BEST
Portability	AVERAGE	AVERAGE	BEST
Accessibility	AVERAGE	AVERAGE	BEST
Ease of use	GOOD	GOOD	GOOD
Complexity	LOW	MEDIUM	HIGH
Maintainability	LOW	MEDIUM	HIGH
Implementation costs	LOW	LOW	MEDIUM
Operational costs	MEDIUM	HIGH	LOW

Pro Tip

Use specific project KPIs to make sure the traffic light section is relevant to business stakeholders.

That's an easy way to outline the chosen solution and build trust.



Follow up questions

Question	Answer

Assumptions

- List the main assumptions,
- of the suggested solution.
- over here.

Key points

- Add key points of the
- Decision in this list

Decision

Capture the decision here.

Why KTDs?

Why KTDs?

We devised them as a process to bring consensus and clarity to complex projects in highly regulated industries.

They are battle tested.

KTDs offer a transformative way to consult, decide and create consensus.

Generally a very well accepted method by teams and clients alike.

Pros and cons.

Trust makes decision making easier.

Teams defining together deliver better.

A decision must be followed through.

Single decision per KTD, don’t cram decisions.

Scope decisions for context, timeframes, budgets.

The collaborative nature of the process is not easy for everyone to follow, usually at the beginning.

From KTD to User Story it’s a long way.

Good KTDs don’t guarantee good delivery.

Don’t use KTD as an excuse for over analysing.

Other standards

Architectural Decision Records

An architecture decision record (ADR) is a document that captures an important architecture decision made along with its context and consequences.

Arc42

arc42 offers a clear, simple and effective structure to document and communicate your software system.

Questions?

What about you?

Take a moment to participate in the survey: https://forms.gle/EZtNBHaY11gri8CE7

What is the process you follow?

Architecture Decision Records?

User Stories + JIRA?

Email chains?

Documented meetings?

Formal signed contracts?

Shared spreadsheets?

Just iterate in sprints and the documentation is the code itself?

Outside your pay range?

What now?

We are open sourcing the KTDs framework.

We have:

Framework documentation. ✅

KTD templates. ✅

A website almost ready. 😬

Github repository. ✅

We need:

Help with translations.

Help with implementation examples.

Help with templates.

Help improving the framework.

Thank you!

Feel free to reach out:

jakala.com

engin.yilmaz@jakala.com

eyilmaz

linkedin.com/in/enyilmaz/

Thank you for joining us!

Stay connected with **Drupal e.V.** for more exciting events, news, and updates!

Also visit our website: **https://drupalcamp.berlin**

Thanks to our sponsors

1x INTERNET

Factorial

agiledrop

Trusted Drupal Teammates

erdfisch

Kraut.Hosting
No Clouds Just Sunshine

dropsolid
EXPERIENCE PLATFORM

arocom.

mittwald.
Hosting neu gedacht

werk21
DRUPAL MADE IN BERLIN

thePHP.cc

TheWeeklyDrop

THE DROP TIMES

talents academy

LOOM

A real example

A real KTD

Problem

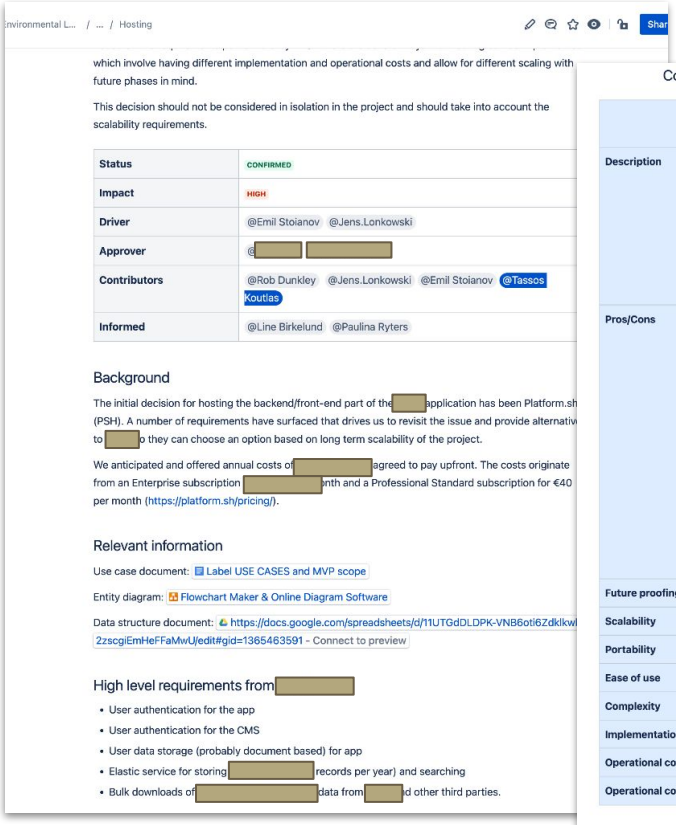
While ⅓ of the way into delivery, new requirements came to light, that challenged the way hosting has been thought of up until that point.

Challenge

Find the best solution to accommodate newest requirements, whilst maintaining platform & project stability

Solution

Run a KTD to decide.



environmental L... / ... / Hosting

which involve having different implementation and operational costs and allow for different scaling with future phases in mind.

This decision should not be considered in isolation in the project and should take into account the scalability requirements.

Status	CONFIRMED
Impact	HIGH
Driver	@Emil Stoianov @Jens.Lonkowski
Approver	@
Contributors	@Rob Dunkley @Jens.Lonkowski @Emil Stoianov @Tassos Koutlas
Informed	@Line Birkelund @Paulina Ryters

Background

The initial decision for hosting the backend/front-end part of the application has been Platform.sh (PSH). A number of requirements have surfaced that drives us to revisit the issue and provide alternativ to they can choose an option based on long term scalability of the project.

We anticipated and offered annual costs of agreed to pay upfront. The costs originate from an Enterprise subscription nth and a Professional Standard subscription for €40 per month (https://platform.sh/pricing/).

Relevant information

Use case document: Label USE CASES and MVP scope

Entity diagram: Flowchart Maker & Online Diagram Software

Data structure document: https://docs.google.com/spreadsheets/d/11UTGdDLDPK-VNB6oti6ZdklkwI 2zscgiEmHeFFaMwU/edit#gid=1365463591 - Connect to preview

High level requirements from

- User authentication for the app
- User authentication for the CMS
- User data storage (probably document based) for app
- Elastic service for storing records per year) and searching
- Bulk downloads of data from d other third parties.

Comparison table

	~~PaaS (PSH)~~	Hybrid (PSH + Third Parties)	Native Cloud (AWS)
Description	~~Use a subscription based Platform as a Service that provides functionality out of the box.~~ ~~PSH does not have a built in authentication service so we're required to go hybrid or native.~~	Use PSH for MVP and on-demand services as needed. E.g. AWS OpenSearch Service, AWS Cognito, AWS DocumentDB	Use DevOps resources to create the needed infrastructure.
Pros/Cons	+ ~~Not having to manage infrastructure~~ − ~~Not able to introduce all services needed~~	+ Good scalability for critical services + Easy development setup + Managed platform so less SLA to worry about − Cost implications as usage increases − Vendor lock-in to particular service	+ Able to accommodate all changing needs of + Native support for user accounts + Ability for Social Login option (SAML) + Less latency + Better MVP cost − Cost implications as usage increases − Pay as you use model − Additional SLA needed
Future proofing	AVERAGE	GOOD	BEST
Scalability	GOOD	GOOD	BEST
Portability	AVERAGE	AVERAGE	BEST
Ease of use	GOOD	GOOD	GOOD
Complexity	LOW	MEDIUM	HIGH
Implementation cost	LOW	LOW	MEDIUM
Operational cost (MVP)	MEDIUM	HIGH	LOW
Operational cost (Long term)	HIGH	HIGH	MEDIUM

Open questions

Key Technical Decisions

Hosting

Created by Tassos Koutlas
Last updated: Nov 09, 2021 by Line Birkelund • 4 min read • 14 people viewed

Based on the requirements put forward by [redacted] there are several ways that hosting can be implemented which involve having different implementation and operational costs and allow for different scaling with future phases in mind.

This decision should not be considered in isolation in the project and should take into account the scalability requirements.

Status	CONFIRMED
Impact	HIGH
Driver	@Emil Stoianov @Jens.Lonkowski
Approver	[redacted]
Contributors	@Rob Dunkley @Jens.Lonkowski @Emil Stoianov @Tassos Koutlas
Informed	@Line Birkelund @Paulina Ryters

Key Technical Decisions

Background

Th[redacted]al decision for hosting the backend/front-end part of the [redacted] application has been [redacted] (PSH). A number of requirements have surfaced that drives us to revisit the issue and provide alternatives [redacted] so they can choose an option based on long term scalability of the project.

We anticipated and offered annual costs of [redacted], agreed to pay upfront. The costs originate from an Enterprise subscription for €1[redacted]per month and a Professional Standard subscription for €40 per mont[redacted]

Relevant information

Use case document: Label USE CASES and MVP scope

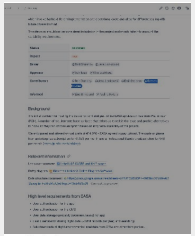
Entity diagram: Flowchart Maker & Online Diagram Software

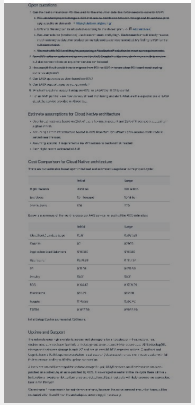
Data structure document: https://docs.google.com/spreadsheets/d/11UTGdDLDPK-VNB6oti6Zdklkwb2 2zscgiEmHeFFaMwU/edit#gid=1365463591 - Connect to preview

High level requirements from [redacted]

- User authentication for the app
- User authentication for the [redacted]
- User data storage (p[redacted] for app [redacted]
- Elastic service for storing flight data (~3.6M records per year) and searching
- Bulk downloads of flight environmental label data from OTAs and other third parties.

Key Technical Decisions

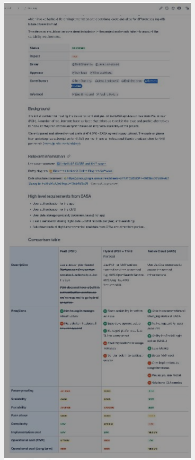


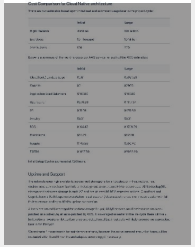


Comparison table

	~~PaaS ([redacted])~~	Hybrid ([redacted] + Third Parties)	Native Cloud (AWS)
Description	~~Use a subscription based~~	Use [redacted] for MVP and on-	Use DevOps resources to
	~~PaaS ([redacted])~~	**Hybrid ([redacted] + Third Parties)**	**Native Cloud (AWS)**
	~~PSH does not have a built in authentication service so we're required to go hybrid or native.~~	DocumentDB	
Pros/Cons	~~Not having to manage infrastructure~~ ~~Not able to introduce all services needed~~	Good scalability for critical services Easy development setup Managed platform so less SLA to worry about Cost implications as usage increases Vendor lock-in to particular service	Able to accommodate all changing needs of [redacted] Native support for user accounts Ability for Social Login option (SAML) Less latency Better MVP cost Cost implications as usage increases Pay as you use model Additional SLA needed
Future proofing	AVERAGE	GOOD	BEST
Scalability	GOOD	GOOD	BEST
Portability	AVERAGE	AVERAGE	BEST
Ease of use	GOOD	GOOD	GOOD
Complexity	LOW	MEDIUM	HIGH
Implementation cost	LOW	LOW	MEDIUM
Operational cost (MVP)	MEDIUM	HIGH	LOW
Operational cost (Long term)	HIGH	HIGH	MEDIUM

Key Technical Decisions





Open questions

1. ~~Can the Elastic instance of [redacted] be used for the amount of data that [redacted] needs to store for MVP?~~
   a. ~~[redacted] standard project storage is 5GB that can be distributed between the app and its services (and upgradeable on demand). → [redacted]~~
   b. ~~CPU and Memory can be allocated according to the chosen plan. →~~ Add services
   c. ~~Requirements for (production) Elasticsearch seem pretty high. Elasticsearch should ideally have us much memory as data. For production multiple nodes are recommended. My feeling is PSH isn't a sufficient choice.~~
   d. ~~**We met with [redacted] and they're proposing a "dedicated" solution to meet our requirements.**~~
2. ~~Does [redacted] offer an authentication service like AWS Cognito? → **No**, they don't have a built in service but also no restrictions so any other service can be used.~~
3. ~~How easy/difficult would it be to migrate from PSH to AWS in future when PSH can't meet scaling demands anymore?~~
4. Can [redacted] approve a solution based on AWS?
5. Can [redacted] support a pay as you go model?
6. What is the optimal account setup on AWS for [redacted]? (for FFW to clarify)
7. For an MVP point of view how we would need monitoring and alert. What are the expectations of [redacted] about the service provided on those two.

Estimate assumptions for Cloud Native architecture

- Cognito estimate was based on 25% of users having an account and 25% of those users accessing in a given month.
- Assuming all core infrastructure hosted in AWS Frankfurt. Cloudfront / CDN ensures static code is served near the user.
- Assuming specific IT requirements like VPN access to backend not needed.
- Each flight record estimated at 4KB

Key Technical Decisions

Cost Comparison for Cloud Native architecture

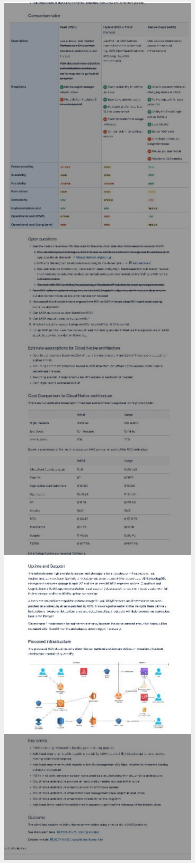
There are two estimates based upon initial load and estimated usage later in the project cycle:

	Initial	Large
[redacted] Records	4 Million	120 Million
End Users	10 Thousand	10 Million
Admin Users	100	250

Below is a summary of the monthly costs per AWS service for each of the AWS estimates:

	Initial	Large
Cloudfront / Lambda Edge	$0.61	$3031.69
Cognito	$0	$2965
Application Load Balancers	$100.80	$100.80
Opensearch	$576.68	$1317.37
S3	$15.69	$470.83
Amplify	$6.11	$6.11
RDS	$164.42	$1508.58
Elasticache	$54.72	$82.08
Fargate	$148.56	$350.40
TOTAL	$1067.59	$9832.86

Key Technical Decisions



Uptime and Support

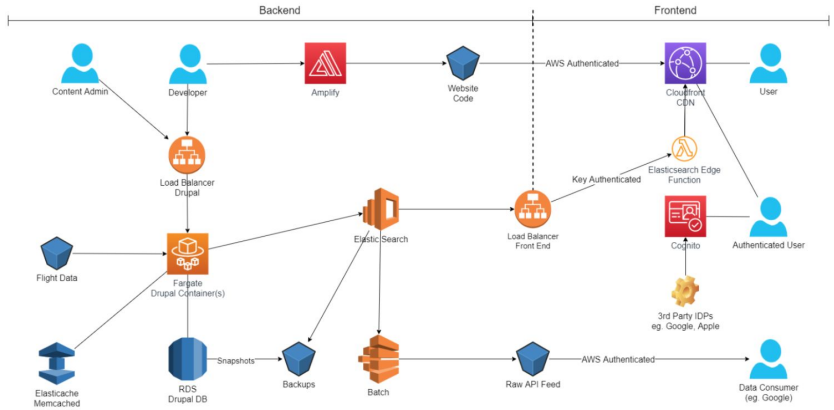
The estimate uses high availability across multiple regions for all production infrastructure. Test environment services have fault tolerant storage but some are not HA to reduce cost. All S3/Backup/File storage and database storage is multi AZ and has at least 99.99% expected uptime. Cloudfront and Cognito have a 99.9% uptime expectation. Elasticsearch / Opensearch runs a 5 or 7 node cluster with full fault tolerance leading to 99.9% uptime expectation.

All services are AWS managed for updates except Drupal, RDS/Database and Elasticsearch are auto-patched on a schedule, all auto-patched by AWS. It is averaged as earlier in the life cycle there will likely be functional tweaks whilst performance and detection of bad input data will likely become more prevalent later in the lifecycle.

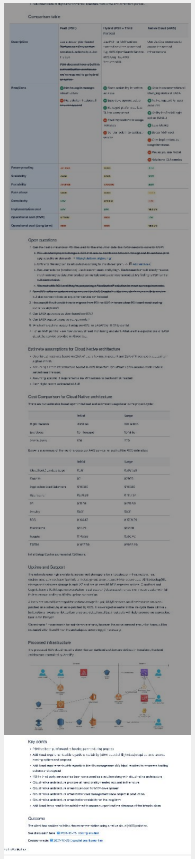
We estimate 4 hours/month for maintenance efforts, however the actual needed amount of hours will be evaluated after 3 and 6 months and adapted accordingly if necessary.

Proposed infrastructure

The proposed AWS cloud model is shown below. Backend architecture defined in Terraform, frontend architecture controlled by Amplify.



Key Technical Decisions



Key points

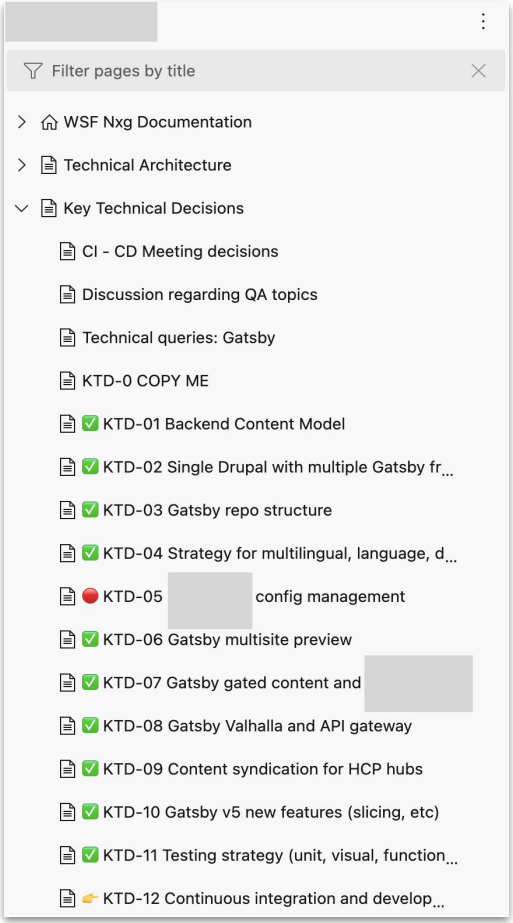
- PSH has been put forward as hosting partner during proposal
- Additional requirements with regards to scalability (40M record of flight data) required to re-assess hosting options and proposal
- Additional requirements with regards to identity management (My trips) required to re-assess hosting options and proposal
- PSH + third party services has been considered as a solution along with cloud native architecture
- Cloud native architecture provides all functionality needed now and in the future
- Cloud native architecture offers lower cost for MVP development
- Cloud native architecture offers better cost management once project in production
- Cloud native architecture offers better scalability for the long term
- Additional tasks need to be established to ensure support and maintenance of the infrastructure

Outcome

The client has decided to follow the recommendation using a native cloud (AWS) solution.

See discussion here: 2021-10-26 Hosting solution

Decision made: 2021-10-29 Expectations & priorities



Solutions outline

Flexible Drupal paragraphs

A solution based on Drupal's contributed module "Paragraph". In this solution there are content components with a well defined content model, so they function really well from a content governance perspective. Also in our solution there will be some flexible content components that can be reused in different situations.

Pros	Cons
Well supported in Drupal world	Content component reusability not out of the box (but supported)
Simplified learning curve	Drupal opiniated approach
Most common way to create content	May require complex JSON API config if there are lots of nesting
Automated garbage collection on content when deleting content	Not in Drupal Core (with huge support however)
Best editorial experience for structured content	

Solutions comparison

	Slicing	Partial hydration	Head API	Script component	TypeGen	Image CDN	Incremental builds	Incremental Clouds
Performance	Very Good	Very Good	Very Good	Very Good	-	Very Good	Very Good	Very Good
Scalability	Very Good	-	-	-	Very Go			
Accessibility	-	Very Good	-	Very Good	-			
Mobile-first	-	Very Good	-	Very Good	-			
Authoring experience	Very Good	-	-	-	-			
Complexity	Average	Good	Very Good	Good	Go			
Portability	Average	Good	Average	Good	Go			
Stability	Unknown	Very Good	Very Good	Very Good	Ve Go			
Maintainability	Good	Good	Very Good	Good	Ve Go			
Decoupled	Average	Good	Average	Average	Av			

Decision

The table bellow summarizes our proposal on what Gatsby new features to be applied on the development of our MVP, based on our initial requirements.

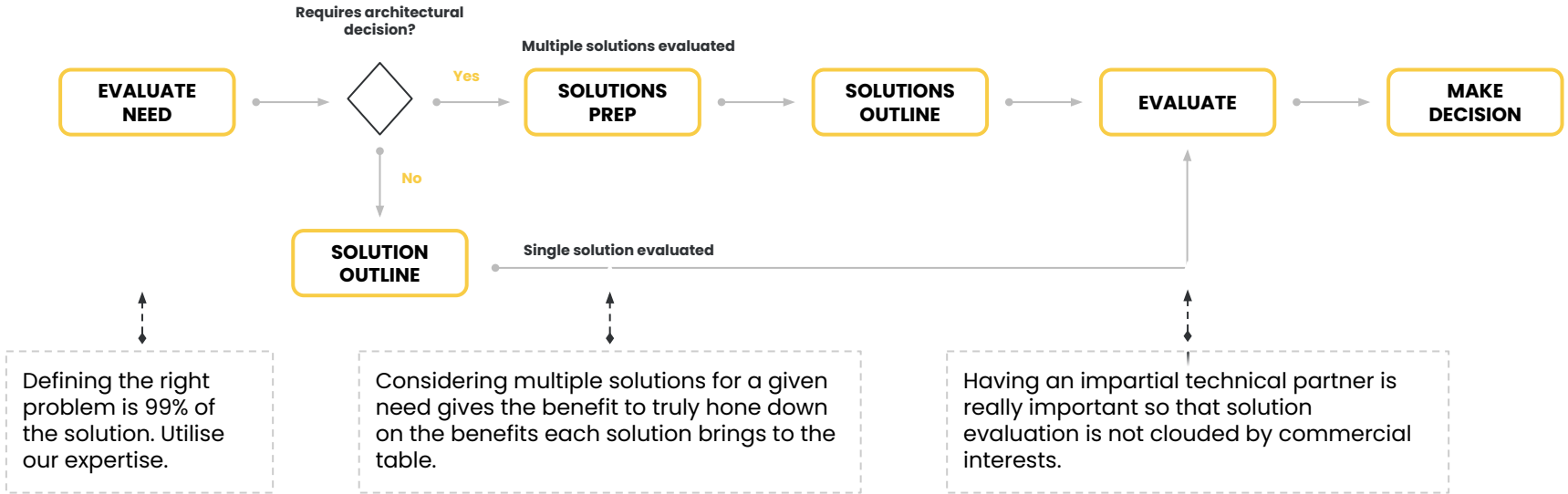
Summary

Feature	Application to MVP	Additional notes
Slicing	yes	Has to be tested in the early stages to make sure it's compatible with Akamai
Partial hydration	yes	
Head API	yes	
Script component	yes	
TypeGen	yes	
Image CDN	yes	
Incremental builds	yes	
Incremental clouds	yes	Is dependent on Fastly.

Workflows

The KTD process

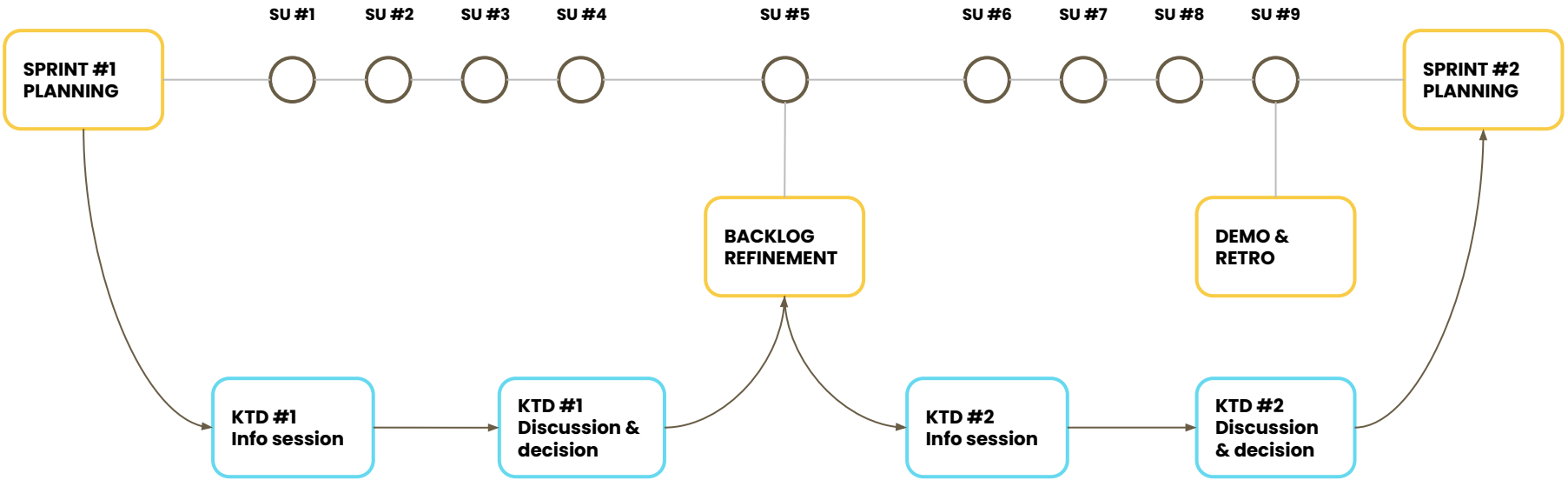
The KTD process is a lightweight but powerful methodology for driving key technical decisions when implementing composable digital experience platforms and digital products. Using it to to drive key technical decisions, organisations are able to accommodate change in dynamic environments and create alignment, consensus and trust with stakeholders.



KTDs in agile sprints

The KTD process can work alongside waterfall or agile methods. For agile methods, the sprint cadence needs to be combined with KTD process so one informs the other during project delivery.

The following diagram showcases how it has been used within agile sprints.



KTDs in waterfall projects

The KTD process can work alongside waterfall or agile methods. For waterfall the workflow is straightforward, make all the decisions prior to technical project delivery.

The following diagram showcases how it has been used within waterfall projects.