Cloudlets Project Overview

Matt Laurence
mattlaurence@gmail.com

A quick note on process

At whichever phase I enter into a project, I like to follow simple, classic design process.

A quick note on process

1. Understand the space, identify the users, empathize with them, learn and research about their goals and challenges to define the problem.
2. Ideate, collaborate, sketch a lot, prototype, test, and iterate until you are pretty sure you solving the problem the right way.
3. Build it and get it in front of people, ready to test and go back to step 2

Use understanding of the user and business needs to make sure we are **solving the right problem**

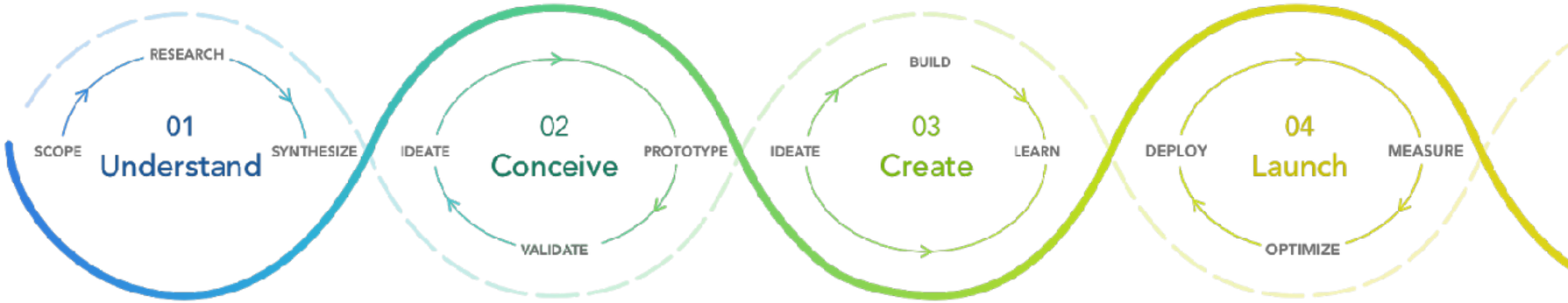
Use what we've learned to make sure we are **solving the problem right**

Build the right product or feature delivering on the original vision, iterating through user validation and quality enforcement

A quick note on process

Of course, we all know it's more complex than that. Each major stage has its own sub-phases, tools, processes, and deliverables, and there can be far more arrows and loops than show up here.

Now I'd like to take you through a real-world sample of my process and work.



FRAME THE PROBLEM

Empathize with users, understand needs, and identify opportunities

- Customer Interviews
- Stakeholder Interviews
- Sales and Support Data
- Competitive Analysis
- Field Studies
- Domain Research
- Workflow Diagrams
- Problem identification
- Personas
- Task Analysis
- Project Briefs
- Requirements Definition
- User Stories
- Journey Maps
- Card Sorting
- Prior Research
- Conceive Workshops

EXPLORE CONCEPTS

Ideate, prototype, and validate concepts with stakeholders and users

- Workflow Diagrams
- Sketching
- Brainstorming
- Prototyping
- Storyboards
- Prototype Testing
- A/B Testing
- Surveys
- Collaborative Tools
- Team Reviews
- Card Sorting
- Contextual Interviews
- Metrics Definition
- Specification Docs
- Initial Wireframes

REFINE THE SOLUTION

Lean UX cycles of building, testing with users and refining

- Usability Testing
- Technical Reviews
- Research Findings
- Finalized Wireframes
- Production Guidelines
- UX Design Handoff
- Technical Review
- Peer Reviews
- Production Code
- User Testing
- Beta Jams
- Quality Assurance
- User Acceptance Testing
- Code Complete
- Product Documentation

LAUNCH & LEARN

Continuously measure ease of use, behavior, and optimize

- Marketing Assets
- Release Notes
- Product Launch
- Governance
- Behavioral Analysis
- Product Surveys
- Voice of the Customers Sessions
- Research Findings
- Usability Testing
- Issue Prioritization
- Backlog Creation
- Documentation Testing
- Go Back to Conceive

History

Akamai was founded as a direct result of Tim Berners-Lee's challenge to combat the "World Wide Wait": Tom Leighton and Danny Lewin created new algorithms and strategies to move content closer to users and minimize traffic latency.

The worldwide network of servers known as "the Edge" has huge power and potential for improving peoples' experience of the Internet.

Background

The first major product I worked on at Akamai was a suite of products known as Cloudlets.

The Cloudlets team typified the deeply engineering-centric Akamai culture: top-notch engineers with little idea what to do with UX.

The business unit leaders, however, identified Cloudlets as a key product that needed to satisfy the new company goal of self-serviceability. Thus, I was made UX Lead on this new-ish product.

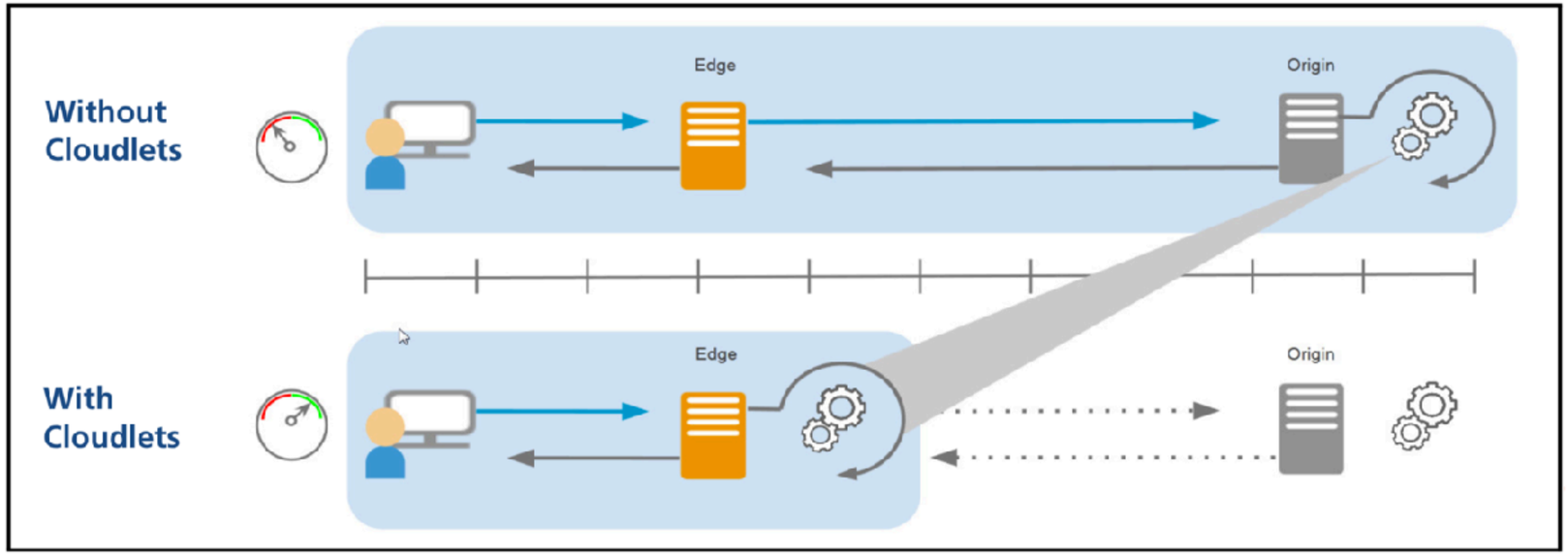
Akamai

Cloudlets

Background

Cloudlets themselves are small, discrete modules designed to bring chunks of logic and functionality out of the origin server and to the Edge, closer to the customer, thus improving performance.

Cloudlets was to meet two of Akamai's company goals: Pushing more power to the Edge, and proving our commitment to the self-serviceability of our tools.



Challenge

When I started there were only two Cloudlets in the ecosystem: Edge Redirector and Forward Rewrite. Both are variations on automating bulk redirection of traffic based on complex conditions and rule sets.

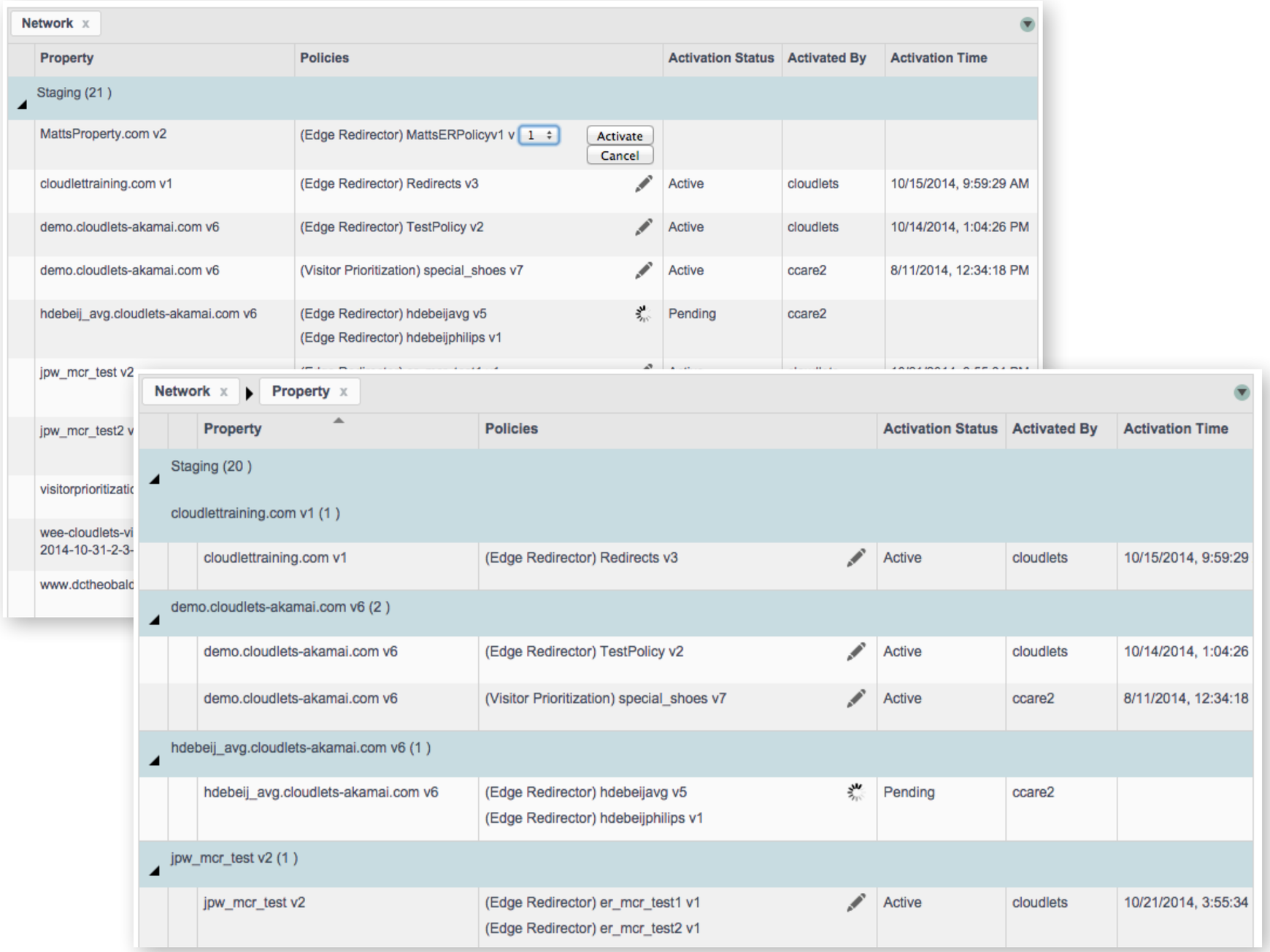
Early Cloudlets customers loved the new, targeted functionality.

But from all reports, they disliked and were confused by the interface.

Challenge

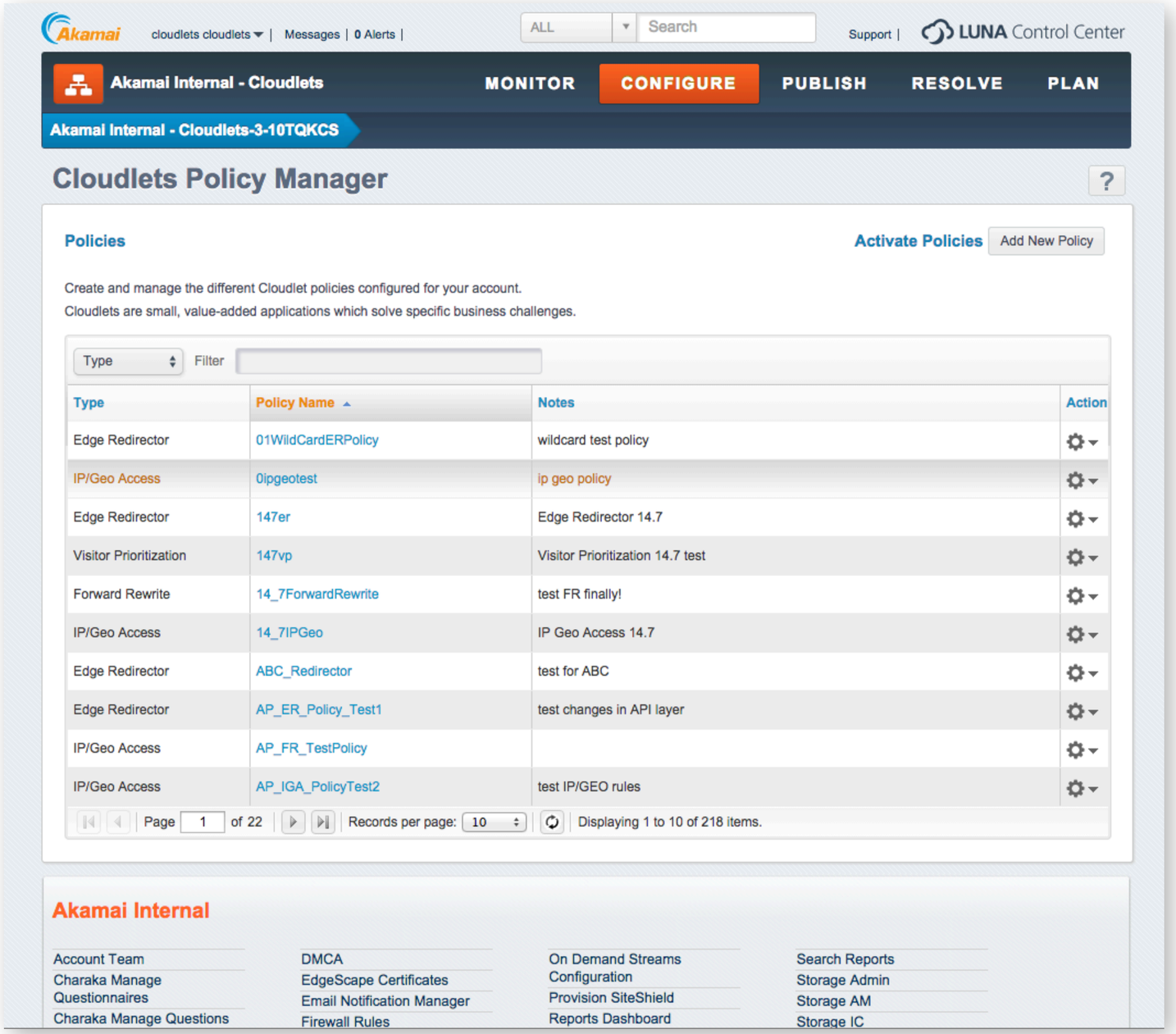
As often happens, the UI had been developed as an afterthought by the engineering team - brilliant software engineers, but not a front-end developer in sight.

Much of it looked and worked very differently from other interfaces on the Akamai ecosystem, and the ways that action and status were represented broke several mental models for Akamai users.



Challenge

Even the simple top-level policy list - the most common page for veteran users - gave almost no indication of status, and even made it difficult for users to filter the list intuitively (as we proved later in testing).



Challenge

To add to the challenge, Cloudlets had already launched, with three more modules being planned, a large team of motivated software engineers, short release deadlines, and a rather reluctant product and sales team.

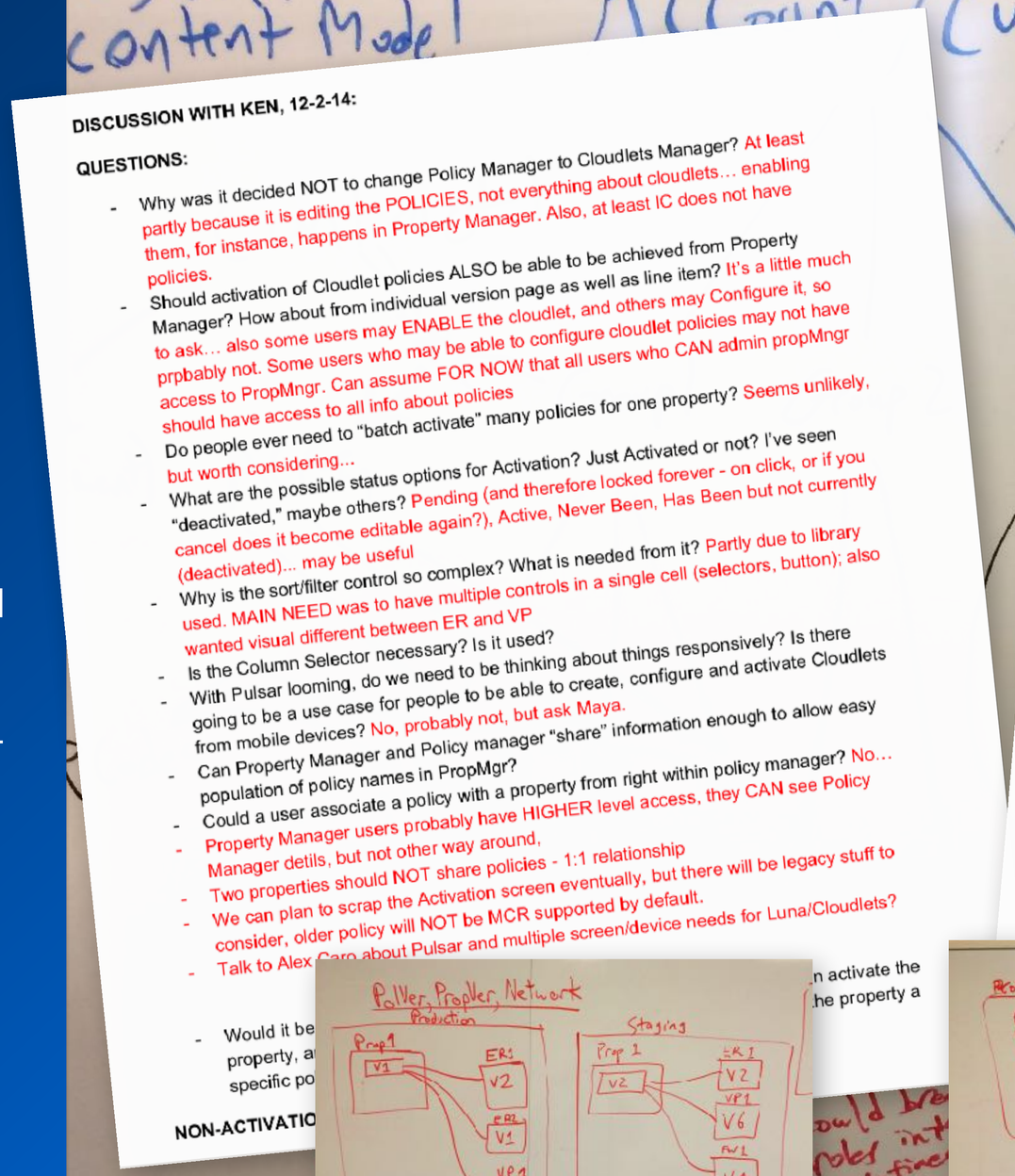
This is where I came in.

Phase 1: Understand

Before and after looking at the Cloudlets UI, I interviewed stakeholders and Professional Services people to understand who the customers were, what problems Cloudlets solved, and to get familiar with the technical domain.

This involved a lot of sketching - and some reworking - of complex relationships and concepts with engineers.

Worth noting: there was no direct customer access at this point, with a very protective product and sales team around customer relationships.



Phase 1: Understand

There were no clearly outlined requirements, so I began codifying my understanding based on these stakeholder interviews and combing through the Jira tickets that served as requirements.

From these I pulled together a better picture of the user roles, use cases, and started to build a narrative and set of understandings and higher level use cases around everything I was learning.

User Roles:

- **Site Administrator**: Akamai customer whose role includes using Property Manager to administer the customer's site. A site administrator would be allowed to **C**reate, **R**ead, **U**pdate, and **D**elete (**CRUD**) policies AND rules.
- **Business Publisher**: Akamai customer who does NOT administer the customer's site, NOR would typically use Property Manager. A business user would be allowed to read and update policies, and create, read, update, delete AND deploy rules within the policies for which the user has update privileges.
- **Business Editor:** Akamai customer who does not deploy rules. Otherwise, identical to the Business "Publisher" role.
- **Viewer**: A user who does NOT administer the customer's site, modify or deploy policies or rules. This user would simply be able to read policies and rules.
- **Customer API client developer**: Akamai customer's programmatic user of the {OPEN} APIs provided for the r
within their own sphere
The intention that this t
policies and rules (i.e.

Any use case that can be perf
Also, any use case that can be
administrator.

POLICY MANAGER USE CASES:

User wants to create a new policy

- From main Policy Manager page, user selects "Create new polic
- Go into what columns/info available, etc. - MOCKUP

User wants to clone a policy

- From the Actions menu, user selects "Clone and edit policy"

User wants to delete a policy

- From the Actions menu, user selects "Delete policy"

User wants to move a policy to a new Property

- From the Actions menu, user selects "Move policy"

User wants to see versions of an existing policy

- From main Policy Manager page, user clicks on the Policy name
Policy versions" from the actions menu

User wants to create a new version

- From the Policy Versions page, user selects "Create new versio

User wants to clone a version

- From the Actions menu, user selects "Clone and edit version"

User wants to delete a version

- From the Actions menu, user selects "Delete version"

User wants to edit version rules

- From the Actions menu, user selects "Edit version rules"
- Go into rule reordering, what columns/info available, etc. - MOCKUP

User wants to Batch upload rules

- From the ???? user selects "Upload rules"

appropriate user type such as Business User)

4. As a Business Editor, I need to navigate through the policies that I can read and/or update:
    a. See all available policies
    b. For a given policy, see all available versions
    c. For a given EDITABLE version, edit all available rules
5. As a Business Editor, I need the ability to create a new version of a policy for which I have update privileges.
    a. I should be able to create a blank new version
    b. I should be able to base the new version based on information from an existing version
6. As a Business Publisher, I need the ability to publish a given version of a policy to the staging and/or production environments (one environment per action).
    a. Once a policy object version has been published to ANY environment, it should be considered read-only.
    b. If a policy object version has been published only to one environment (e.g. staging), I still need to be able to publish the SAME version to the other environment (that is, production in this example)
7. As a Business Editor, I need to be able to name policies and rules (though these fields are NOT required). Versions of a policy should have a short description field to help me keep track of my changes.

Edge Redirector

1. As a Business Editor, within an editable version of a policy, I need to be able to create a new redirect rule and update or delete an existing redirect rule
2. As a Business Editor, I need a redirect rule to be able to match on the incoming:
    a. FULL URL, OR
    b. The AND of the following clauses (as appropriate):
        i. hostname of the request
        ii. partial or full file path (i.e. /products/sale/*)
        iii. one or more file extension (*.jpg, or "*.jpg *.gif")
        iv. protocol
        v. any one query string parameter
        vi. country
        vii. continent
        viii. region
        ix. cookie
3. As a Business Editor, for a given redirect rule I need to be able to specify the URL that is handed back as the value of the 'Location' HTTP response header. Additionally, I may:
    a. Specify that the redirect should include the entire query string (if any) present in the original request
    b. Whether it is a Temporary or Permanent redirect (i.e. 301 vs 302)
4. As a Business Editor, I need to be able to schedule start and end times for any given

Phase 1: Understand

During this initial round of research I continued to expand and refine my overall discovery document, my “single source of truth” for my own efforts.

This contained all of my understanding about the product, users, and ecosystem, as well as the ever more refined use cases, for collaboration and feedback with many sometimes remote team members.

Purpose and Scope of this Document

This is a functional specification for the Cloudlets Manager GUI in the Akamai Luna Portal. It is meant only to explain the concepts and processes relevant to each Luna user's experience for configuring, activating, and reporting on Cloudlets. Discussions of the underlying implementations of Cloudlet logic, data representations and storage, APIs, Cloudlet type-specific details (e.g. Edge Redirector, Visitor Prioritization), integration with Ghost, and contract entitlements, etc. are out-of-scope as are detailed look-and-feel mockups.

Background

From Winter 2013 through Fall 2014, the evolution of the graphical user interface (GUI) for Cloudlets Manager led to a number of usability issues because it evolved through a process having some critical shortcomings:

- There was no document to unify the series of offline and online design discussions (email, JIRA tickets, and wiki pages), so it was easy to lose track of why certain decisions were made or what exactly was envisioned. Furthermore, we lacked a common understanding across the core team of the roles, user stories, and use-cases that we needed to support.
- The bulk of the GUI was implemented with little or no guidance from UX experts, so standard usability features (e.g. proper descriptions and tooltips) were sometimes missed. Another effect of this is that the look-and-feel diverged from that of both Property Manager and the guidelines of the Akamai User Experience (AUE aka UXD) group.

In late Fall 2014, a number of factors have led us to formalize and improve the GUI:

- Akamai customers began to use and give feedback on Cloudlets Manager resulting in a spike in bug reports and requests for usability enhancements.
- Actual and projected customer adoption rates have jumped
- Product management started planning for the addition of many more Cloudlets throughout the coming year.
- An upcoming back-end improvement (expected in Portal 15.1 in Jan 2015) called “Multiple Content Realm” (that will allow for independently activating Cloudlet policies instead of in groups) will allow us to redesign the GUI for our activation process to be more similar to how it is done in Property Manager.

Related Documentation

Document	Link	Comments
Nimbus Cloudlets high-level overview	https://gsd.akamai.com/wiki/display/CLOUD/NIMBUS+Cloudlets	
NIMBUS Cloudlets FAQ	https://gsd.akamai.com/wiki/display/CLOUD/NIMBUS+Cloudlets+FAQ	
Policy activation workflow (current as of 11-24-14)	https://gsd.akamai.com/wiki/display/~dtheobal/2014/02/12/NIMBUS+Policy+Activation+Workflow	Redesign of this flow is starting of 12-10-14, currently planned for release in 15.2
Cloudlets open JIRA tickets for Bugs	https://gsd.akamai.com/jira/browse/WEEPER	
Cloudlets open JIRA tickets for Features and Enhancements	https://gsd.ak	
Technical methodology for building TAPIOCA-based Cloudlets	https://gsd.a	
TAPIOCA Spec	https://docs	
Cloudlet tracking, marketing and other links	https://gs	
Cloudlets Service and Support Wiki	https://a	
Public site overview of Cloudlets	http://h	
Link to regex func spec, newer cloudlets, etc	https	

NIMBUS Cloudlets User Stories and Use Cases

User wants to create a new policy

1. User clicks the Create Policy control
2. System presents the Create Policy modal dialog
3. User enters required information:
   a. Cloudlet type
   b. Policy Name
   c. Policy Notes
   d. Property Binding (optional)
4. System enables Create Policy button
5. User clicks Create Policy Button
6. System initiates server commands to create new Policy
7. System displays progress spinner with function description “Creating New Policy...”
8. System removes modal dialog and takes user to Policy Details page for the new Po
   a. ALT: System removes modal dialog and presents failure message

User wants to clone a policy

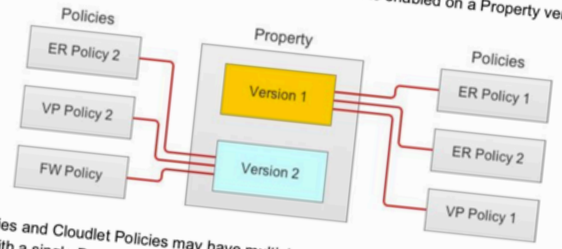
1. User selects Clone Policy from the Actions menu
2. System presents the Clone Policy modal dialog with enabled Clone Policy button
3. User optionally updates information:
   a. Policy Name
   b. Policy Notes
   c. Property Binding
4. User clicks Clone Policy Button
5. System initiates server commands to Clone the Policy
6. System displays progress spinner with function description “Cloning Policy...”
7. System removes modal dialog and takes user to Policy Details page for the cloned Polic
   a. ALT: System removes modal dialog and presents failure message

User wants to delete a policy

1. User selects Delete Policy from the Actions menu
2. System presents the Delete Policy modal dialog with enabled Delete Policy button
3. User optionally updates information:
   a. Policy Name
   b. Policy Notes
   c. Property Binding
4. User clicks Clone Policy Button
5. System initiates server commands to Clone the Policy
6. System displays progress spinner with function description “Cloning Policy...”
7. System removes modal dialog and takes user to Policy Details page for the cloned Policy
   a. ALT: System removes modal dialog and presents failure message

Cloudlet Policies have a many-to-one relationship with the Properties they support, as illustrated in the image above. Any given Property may have several different types of Cloudlet associated with it, and it may also have multiple instances of a single type of Cloudlet. However, each Cloudlet Policy and its versions may be associated with only a single Property

Cloudlet policies are created and edited in Cloudlets Policy Manager, then added to a **specific Property version** through Property Manager; this binds the policy to only that Property. A Cloudlet Policy must be specified at the time when a Cloudlet behavior is enabled on a Property version.

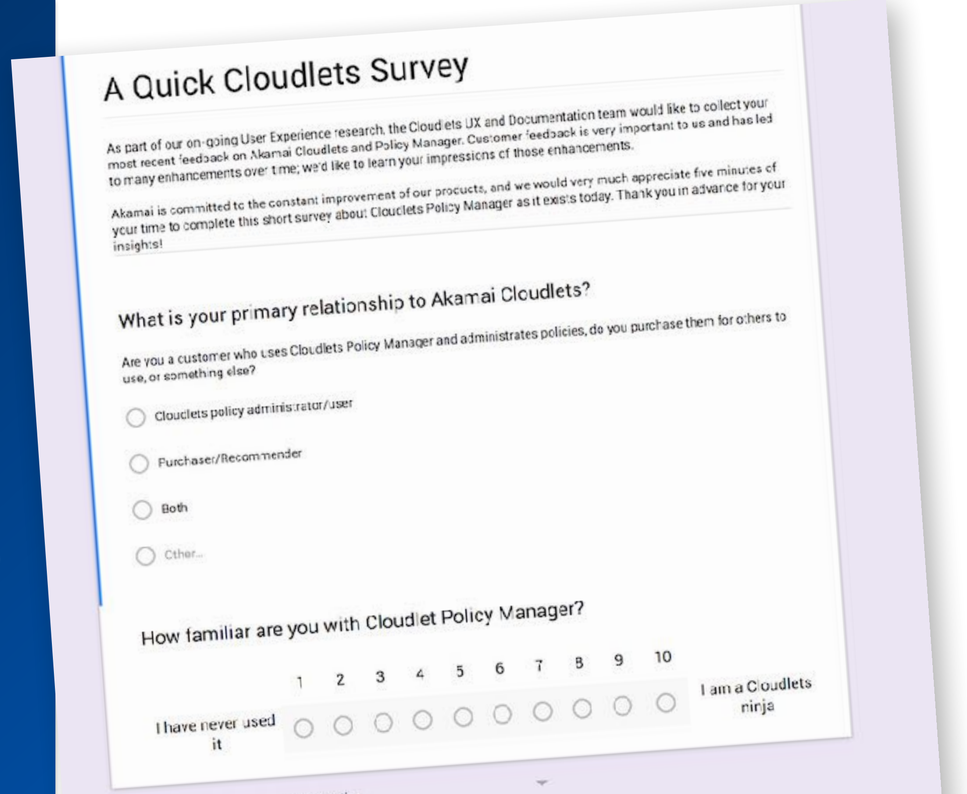


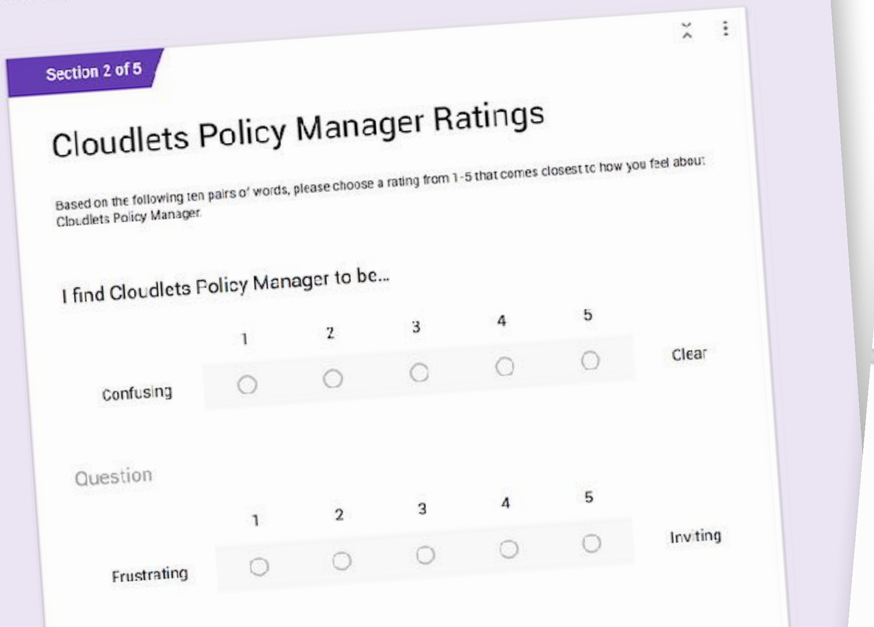
Both Properties and Cloudlet Policies may have multiple versions. Each Cloudlet Policy may only be associated with a single Property, but any individual Cloudlet **Policy ver**

Phase 1: Understand

In this time, through a series of face-to-face discussions and a slow building of credibility and, I also managed to get approval to send out a baseline survey to both internal and external Cloudlets users.

The quantitative aspect of this survey was based around a statistical rating tool that I created with my colleague Carolyn Griffel, which we called Web Experience Usability Dyads.





Cloudlets Survey 6-15 Qualitative feedback

This document contains all of the free-text responses to the final survey question:

"Is there anything else you'd like to share with us about Cloudlets Policy Manager or your experience with it? Please let us know any thoughts, ideas, feedback, or clarification you have about Cloudlets here."

FROM CUSTOMERS:

If I can import redirects, I should be able to export them as well.

In rule definition, I want more booleans! There is no ability to OR! **(No need - De Morgan's law)**

I also don't like having to put a fully qualified URL in the redirect

---

more documentation, examples, best practices

"The whole approach of how it's implemented is backwards.

Why can't I have one interface to update rules, and activate in one place?

Seems like technical barriers drove design.

Biggest problem: it's buggy and unpredictable in usage. That is at least quite troublesome for a PROD system

Roles are not good enough.

Admin cannot use GUI to delete policies

Not flexible enough to restrict policies for users.

Please add How-To for Cloudlets Policy manager with explanation of Regex usage (variables, syntax, modifiers,....).

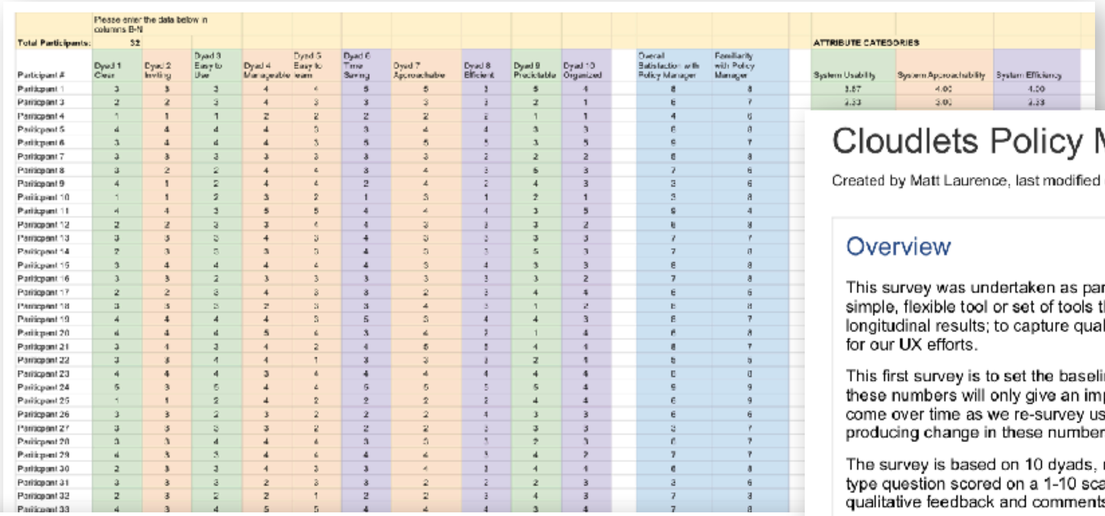
The relationship between a request, property manager config and cloudlet policies can be hard to trace. Adding a response header with policy ID on staging (at least) would be helpful. As would anything that allows testing of behaviour

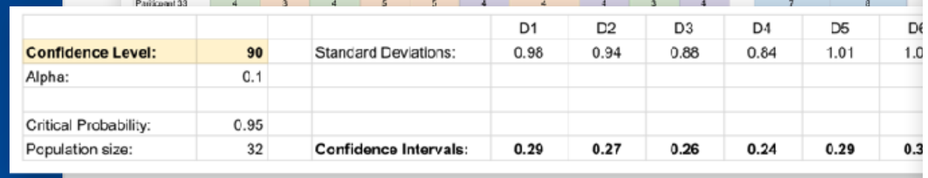
---

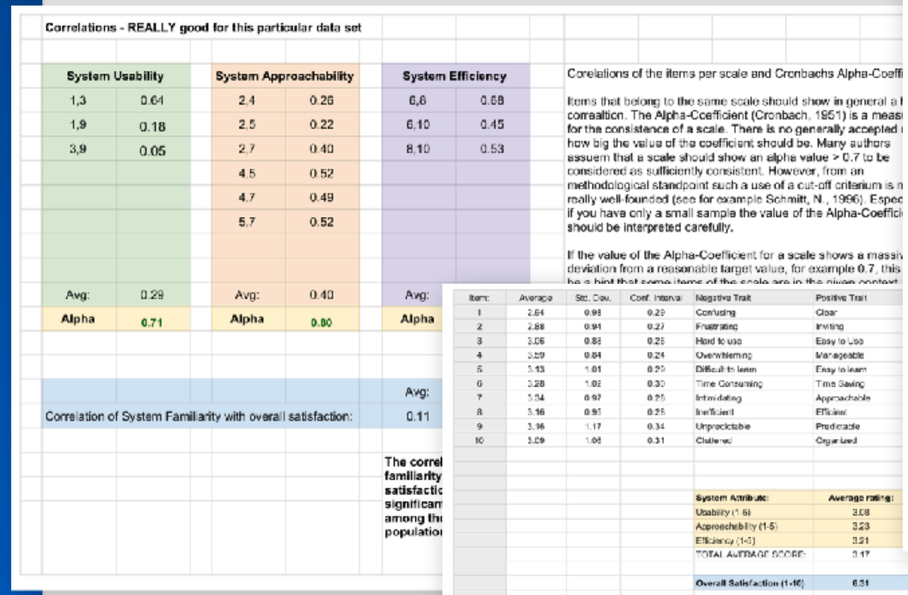
Phase 1: Understand

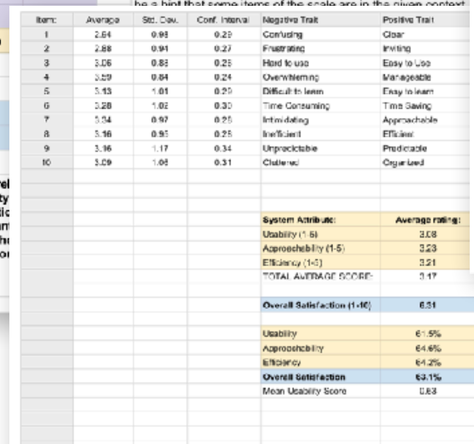
Survey results were crunched using my Excel tool (as validated by a statistician and data scientist) to generate quantitative rankings of system usability, approachability, and efficiency for both internal and customer users.

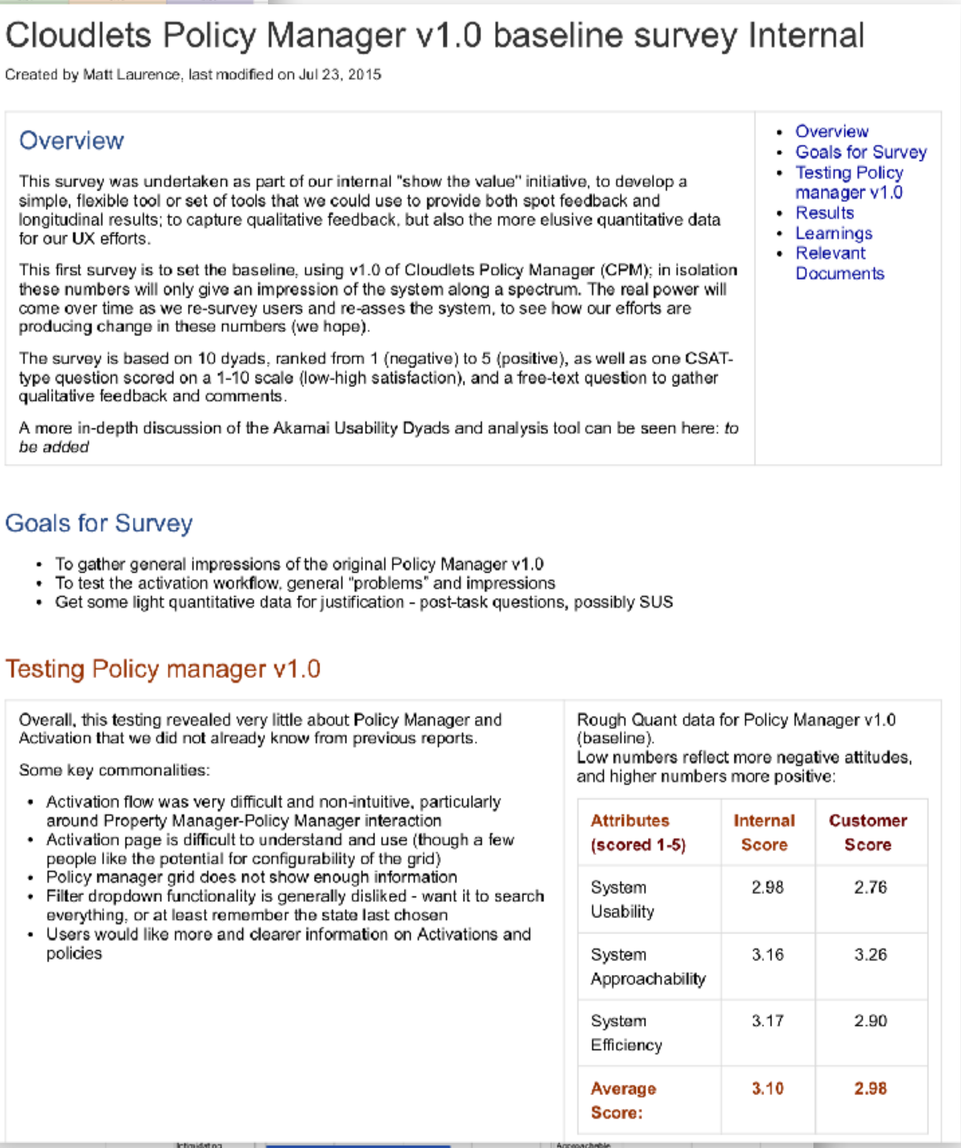
We also gathered qualitative feedback as part of the survey process, which provided perhaps even more compelling insights











Cloudlets Policy Manager v1.0 baseline survey Internal

Created by Matt Laurence, last modified on Jul 23, 2015

Overview

This survey was undertaken as part of our internal "show the value" initiative, to develop a simple, flexible tool or set of tools that we could use to provide both spot feedback and longitudinal results; to capture qualitative feedback, but also the more elusive quantitative data for our UX efforts.

This first survey is to set the baseline, using v1.0 of Cloudlets Policy Manager (CPM); in isolation these numbers will only give an impression of the system along a spectrum. The real power will come over time as we re-survey users and re-asses the system, to see how our efforts are producing change in these numbers (we hope).

The survey is based on 10 dyads, ranked from 1 (negative) to 5 (positive), as well as one CSAT-type question scored on a 1-10 scale (low-high satisfaction), and a free-text question to gather qualitative feedback and comments.

A more in-depth discussion of the Akamai Usability Dyads and analysis tool can be seen here: *to be added*

- Overview
- Goals for Survey
- Testing Policy manager v1.0
- Results
- Learnings
- Relevant Documents

Goals for Survey

- To gather general impressions of the original Policy Manager v1.0
- To test the activation workflow, general "problems" and impressions
- Get some light quantitative data for justification - post-task questions, possibly SUS

Testing Policy manager v1.0

Overall, this testing revealed very little about Policy Manager and Activation that we did not already know from previous reports.

Some key commonalities:

- Activation flow was very difficult and non-intuitive, particularly around Property Manager-Policy Manager interaction
- Activation page is difficult to understand and use (though a few people like the potential for configurability of the grid)
- Policy manager grid does not show enough information
- Filter dropdown functionality is generally disliked - want it to search everything, or at least remember the state last chosen
- Users would like more and clearer information on Activations and policies

Rough Quant data for Policy Manager v1.0 (baseline).
Low numbers reflect more negative attitudes, and higher numbers more positive:

Attributes (scored 1-5)	Internal Score	Customer Score
System Usability	2.98	2.76
System Approachability	3.16	3.26
System Efficiency	3.17	2.90
Average Score:	**3.10**	**2.98**

Phase 1: Understand

I presented results from the survey to explain the methodology and help provide context for recommendations.

While the quant numbers were valuable as a baseline metric, the qualitative feedback and quotes were the most impactful with the team.

User Research

Baseline testing:

- Interviews with SMEs and Customers
  - Tested certain tasks as well as qualitative respons
  - Gave valuable insight for CPM redesigns
- Baseline survey done Pre-15.4

WEE Dyad scale+

System Usability

Confusing	Clear
Frustrating	Inviting
Hard to use	Easy to Use
Overwhelming	Manageable
Difficult to learn	Easy to learn
Time Consuming	Time Saving
Intimidating	Approachable
Inefficient	Efficient
Unpredictable	Predictable
Cluttered	Organi…

Survey Results June 2015

	SME	Customer
System Usability 1-5	2.98	2.76
System Approachability 1-5	3.16	3.26
System Efficiency 1-5	3.17	2.90
Overall Satisfaction 1-10	6.05	5.18

Survey Results June 2015

Based on SUS, 68% is average usability

	SME	Customer
Usability 1-5	59.6%	55.2%
Approachability 1-5	63.2%	65.2%
Efficiency 1-5	63.4%	58.0%
Satisfaction 1-10	60%	51.8%

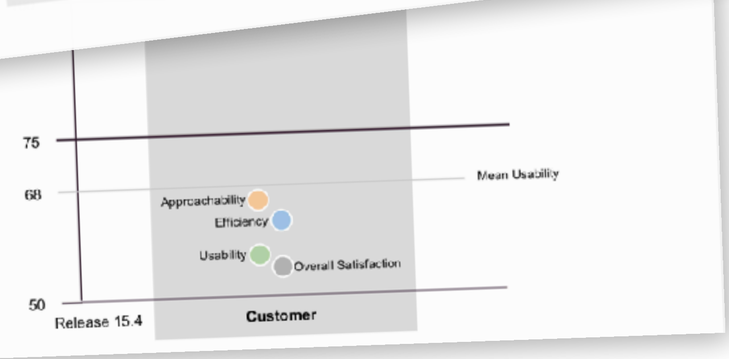
User Feedback

Some negatives:

"It's buggy and unpredictable in usage. That is at least quite troublesome for a Production system"

"Seems like technical barriers drove design."

"The fact that policies have to exist and cannot be deleted is a severe issue with us and not very scalable."



Phase 1: Understand

I then created UI “spec” documents for each of the two specific target areas that needed addressing: the Policy Manager interface and the Activation workflow.

These documents help me to clarify the goals and challenges in my own terms, collaborate with team members, and provide the new “single source of truth” for each discrete aspect of functionality.

Cloudlets Policy Activation Workflow

Overview
Relevant documentation and tickets
Important Points in the New Design:
Workflow and Wireframes
LInk to Wireframes in Confluence
Cloudlet Activation Lifecycle
Configuration and Activation Steps current (as of 1-21-15):
Scope
Requirements/Assumptions
Configuration and Activation Steps PROPOSED phase 1:
Proposed Page & Grid Designs
Policy List (Main Page):
Clone Policy modal:
Policy Details page:
Policy Details page - Activation Modal:
Activation History Tab, Policy Details page:
Version Details page:
Property Manager interaction
Property Manager - Cloudlet behavior with a policy selector:
Property Manager - Policy selection overlay:
Documentation considerations

Overview

Customers have complained that the workflow for activating Cloudlets policies is complex and time consuming. While part of that is the lengthy UMP “push” required for some Cloudlets (ER, FR, and IG beta) which is on the roadmap to be addressed in a future release, part of it is the flow through the GUI.

As of 15.2, policy activation is done via the Activations grid in the Cloudlets Manager application. This grid is organized by network (Staging v Production) and then each row represents a property version with its associated policies (one or many). The user then can select a set of versions for each policy on a property and click the “Activate” button which will submit the version set.

Workflow and Wireframes

LInk to Wireframes in Confluence

Cloudlet Activation Lifecycle

There are three main parts to the new activation lifecycle:

1. Reserving a policy for a property. This can be done either by:
   a. (in Prop Mgr) Adding an unreserved policy to a behavior instar
   verison.
   b. (in Cloudlets Mgr) For a policy that is not yet reserved on an ac
   “locked”), it can be assigned/reassigned to another property.
2. (Property Activation) Activating a Property version containing Cloudlet b
   each reference a particular Cloudlet policy.
   a. Requires a user with activation permission in Property Manager.
   b. The Property version can be activated on one or many networks,
   Production.
   c. The only policies that can be associated with the Property version
   have not already been associated with a version of a different Prop
   which exist in or under the same group.

(Policy Activation) Activating a particular version of each policy on a netwo
ong as it has been reserved for a Property.

a. Requires a user with activation permission in Cloudlets Manager for
   type.
b. The policy version can be selected from any that have been created

ion and Activation Steps current (as of 1-21-15):

to typos in the policy name that must be typed in the property behavior
ion screen is separate from main policy grid and confusing.
ng sets of policy versions per property version is awkward.
n status information is very limited.
no view into the activation history.

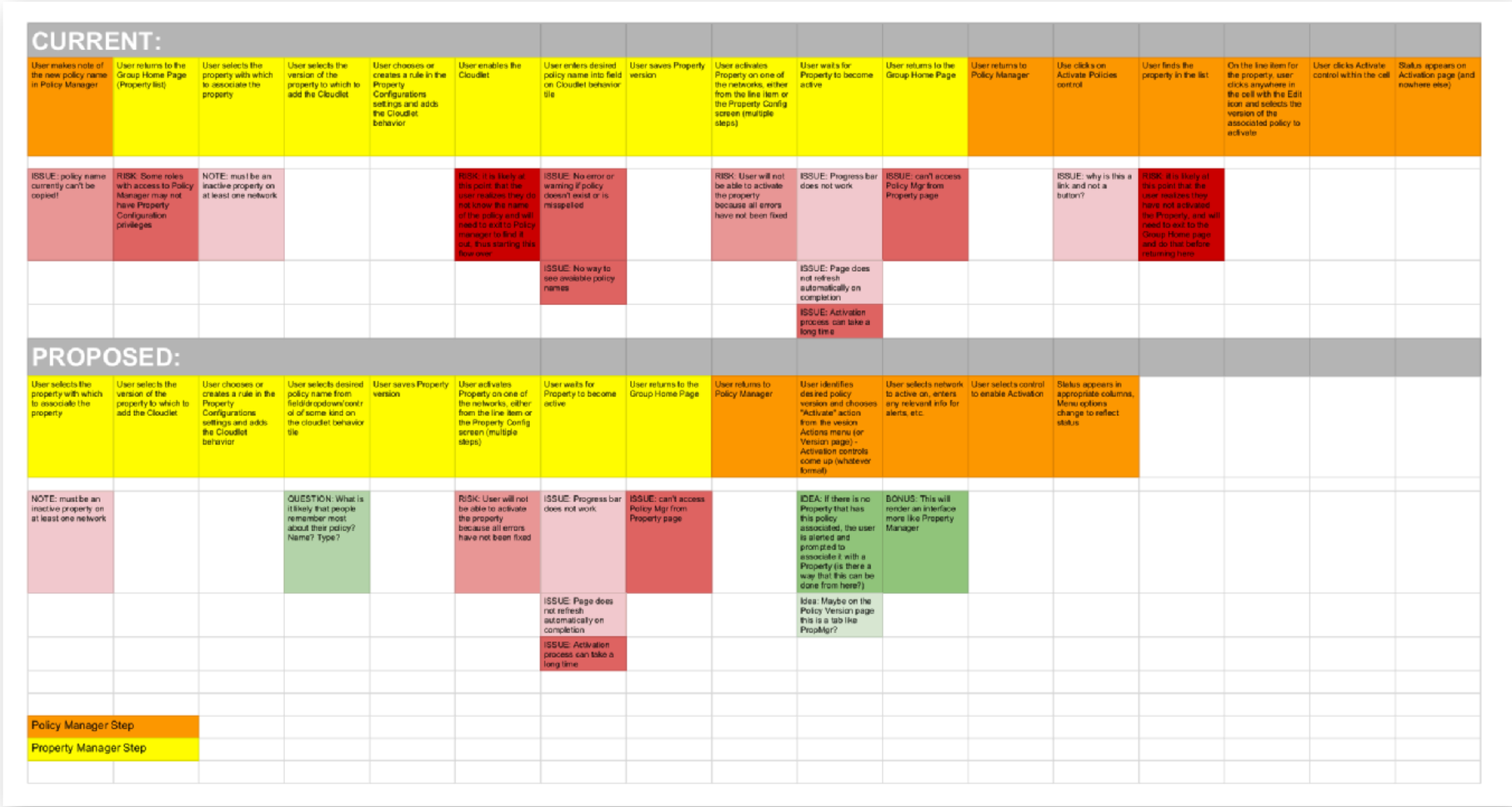
es that the target Property exists, and that the user has privileges to ac
nager and Policy Manager. In red below are pain points or things we k
he redesign in 15.4)

ets Policy Manager from the Megamenu
In Policy Manager, create a **Policy**
- Add **rules** to a **version** of that policy

Phase 1: Understand

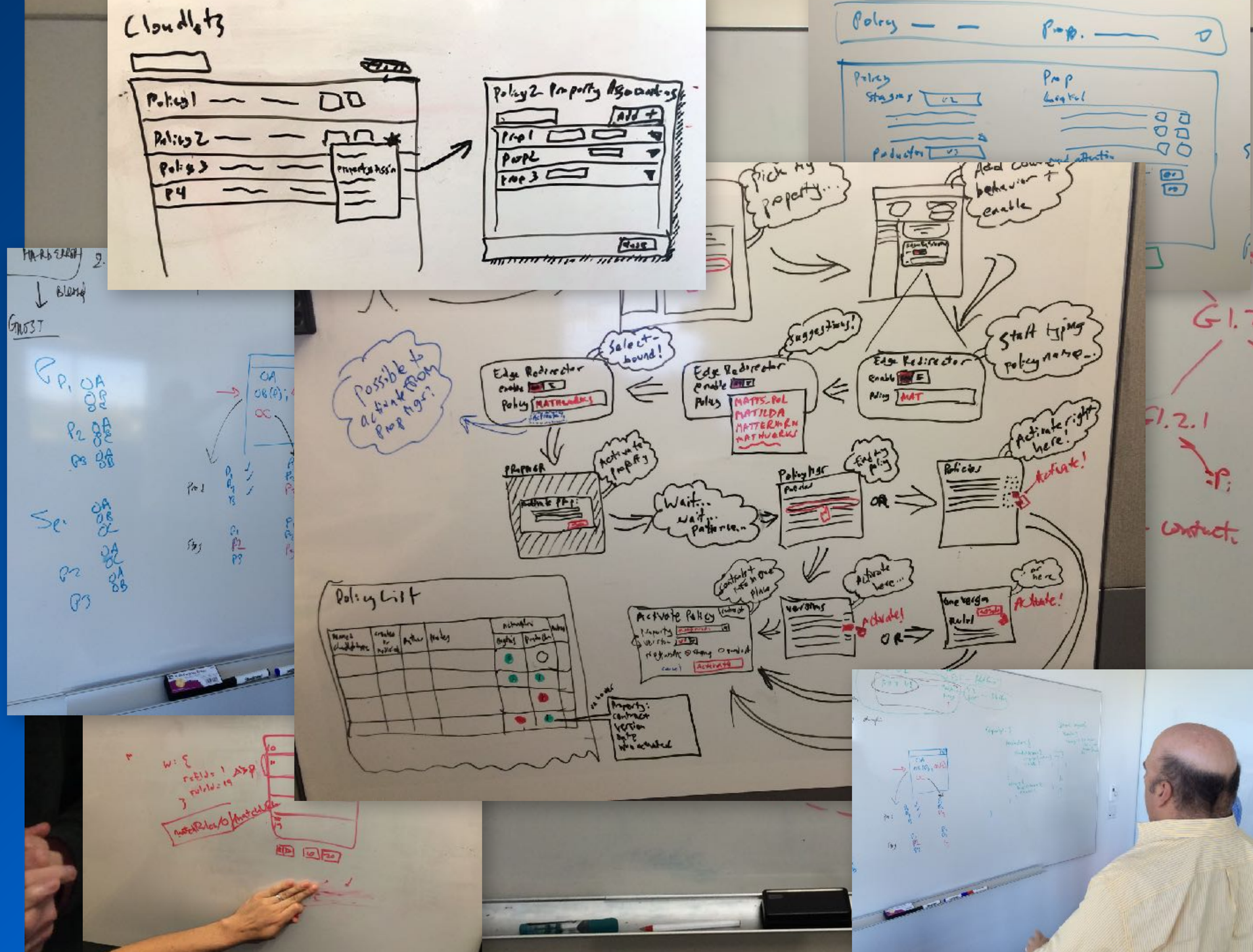
Using feedback from our internal users and support team I created a simple journey map through the existing activation process, highlighting pain points, risks, notes around the workflow, and steps therein.

This screen shows a somewhat later version in which an alternate, more condensed workflow is proposed, within the limitations of the system.



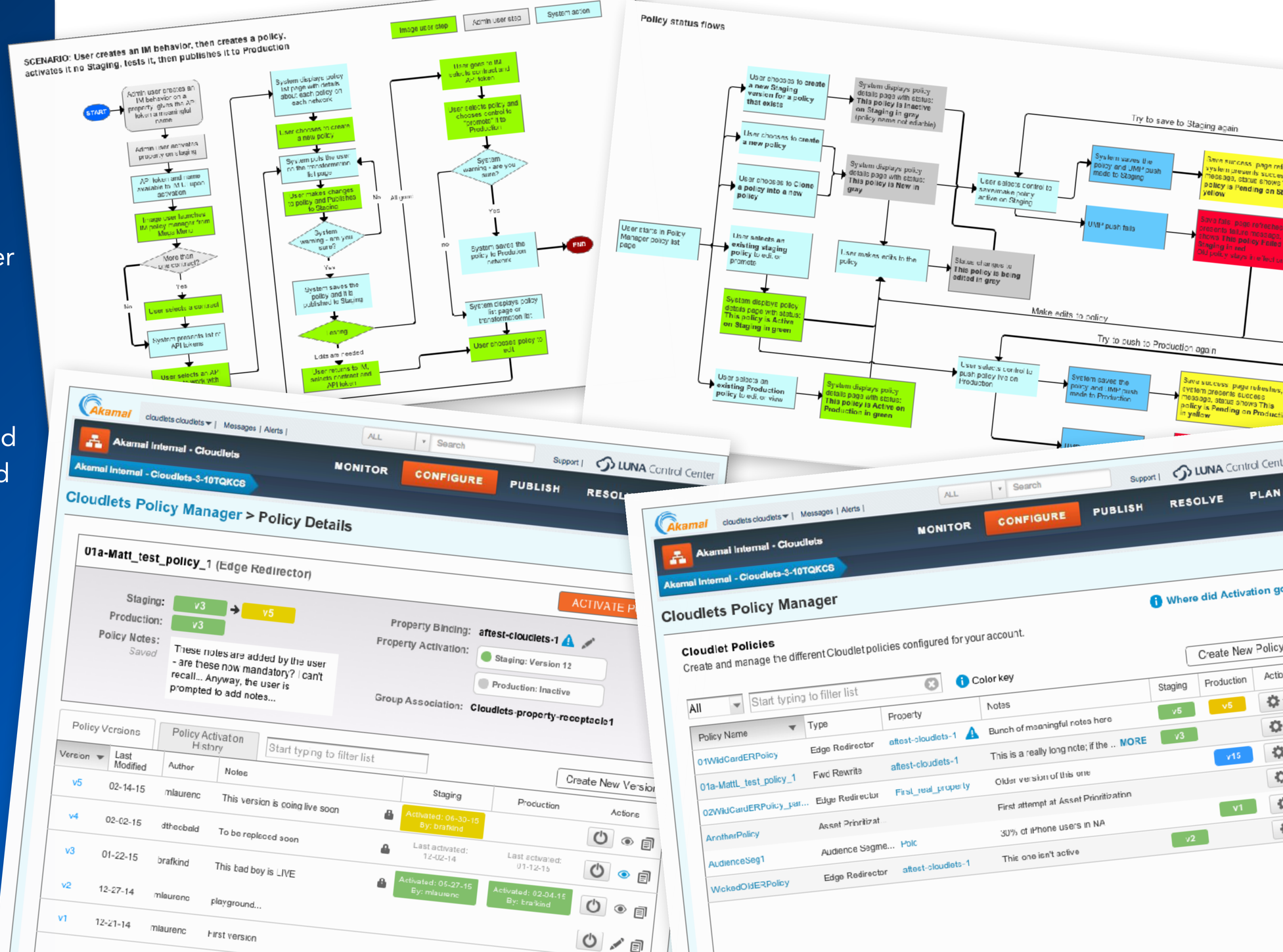
Phase 2: Conceive

Once we had agreement on the general flows, I got to work with the product owner, software architect, and engineers, doing a lot more sketching to start to put this all in context of the larger Akamai ecosystem, and trying out different ways to minimize our customers' frustration.



Phase 2: Conceive

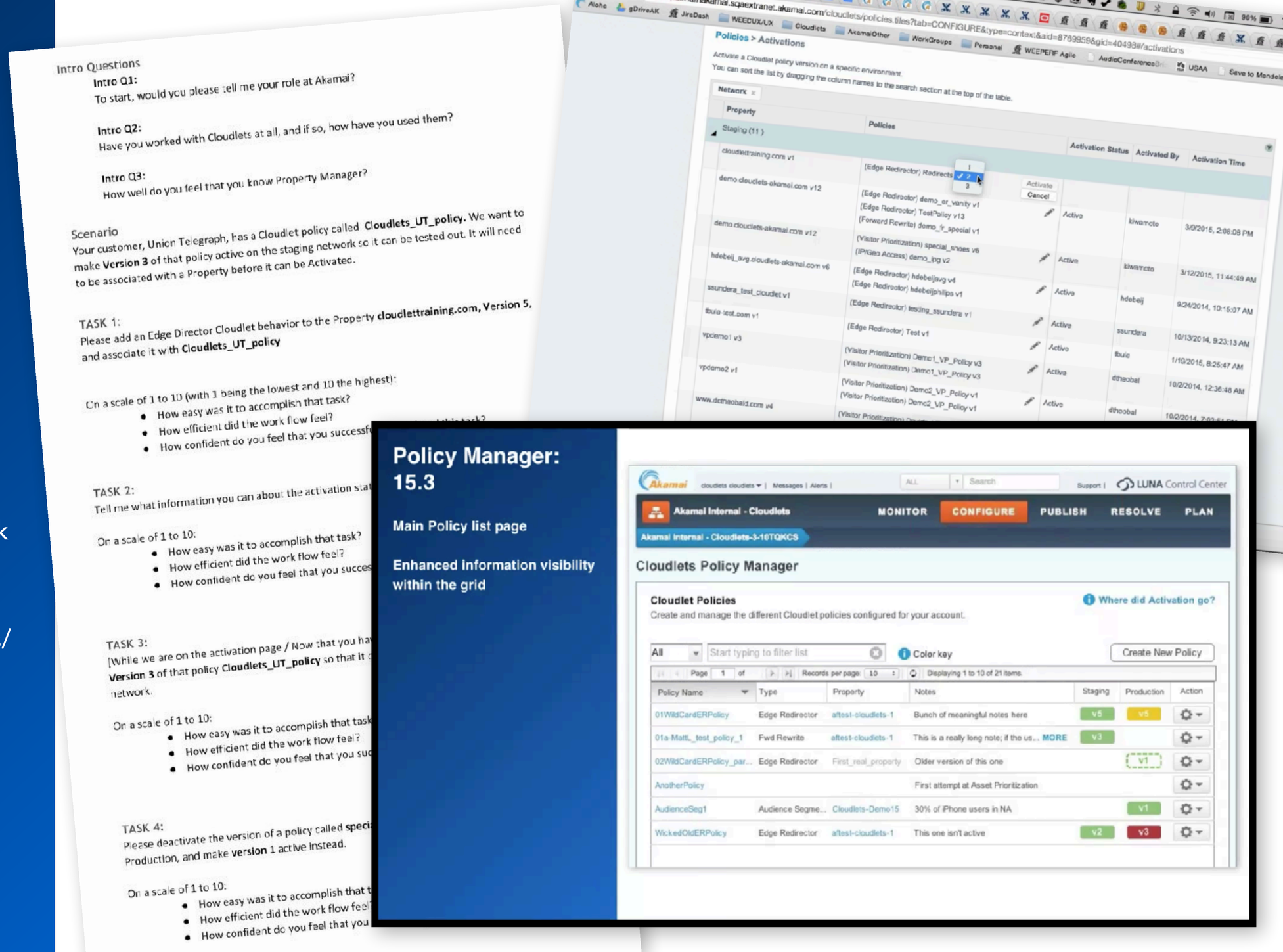
As the definition became clearer and the solutions more defined through iterative reviews and work sessions, I created more detailed workflows and wireframes for the use cases and UI elements, to be incorporated into my spec document as a continued collaboration tool with the developers.



Phase 2: Conceive

During this phase I also ran a usability test on the original UI and workflows with internal professional services participants, to uncover particularly sticky areas of the workflow and UI.

In addition to seeing them work with the original UI, at the end of those sessions I also showed them the in-process wireframes/ mockups to get their reactions and input before we began building anything.



Phase 3: Create

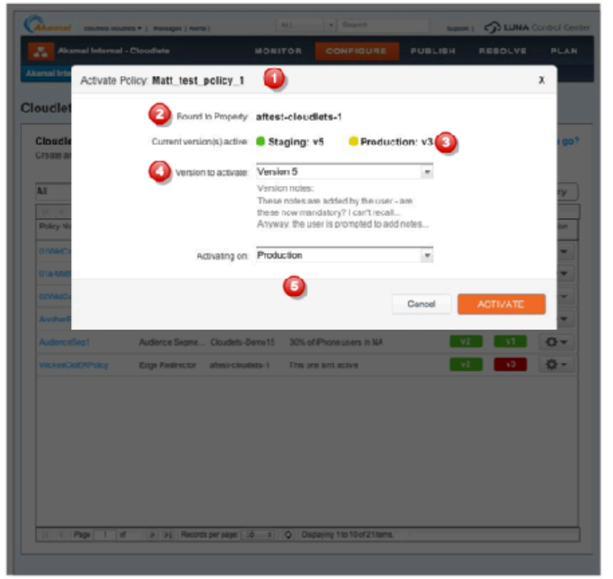
Over time after many rounds of collaboration and iteration, I created an annotated set of wireframes from which the developers could use for guidance during their build process.

During the build process I spent a great deal of time sitting elbow to elbow (or webex to webex) with the developers inspecting elements in the browser and tweaking CSS and Javascript.

Activation Panel

When admins choose to activate a Policy, they use this panel to do so

1. Activation header clearly shows the policy name
2. Property binding is shown for reference, but is not editable from this panel
3. The versions that are currently active will display here so the user has a clear view of what is going to be deactivated through this action, if anything.
4. If the user selects to activate an item from the line item or from the Version page, this dropdown is pre-populated with that version, otherwise it is populated with the most recent version number.
   a. The version notes for the selected version should show here (SME feedback) so that the user has full insight into the details of the new version they are activating
5. SMEs have very strongly indicated that Activation Notes should be allowed here, both for their inherent usefulness and for parity with Property Manager.

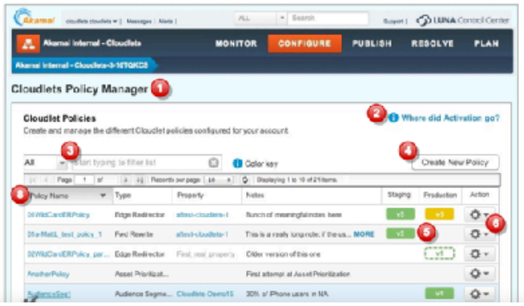


Annotation	Wireframe/Mockup

Main Policy List

This is the main page of Cloudlets Policy Manager

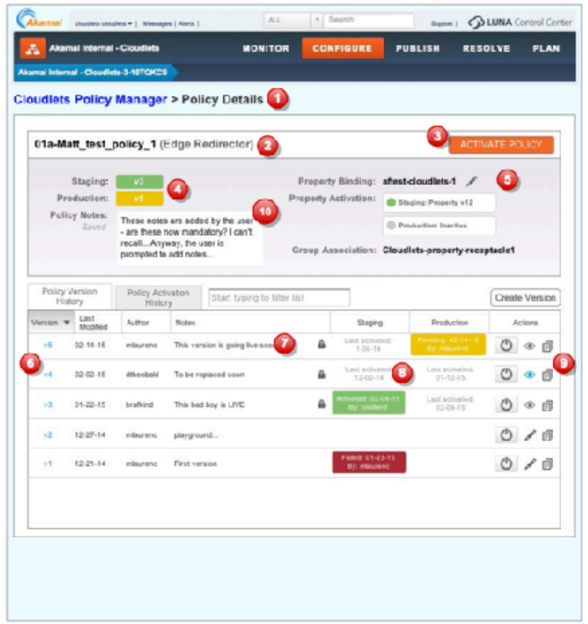
1. The Cloudlets Policy Manager title line now functions as a working breadcrumb
2. Activations link is replaced by a link to reveal bubble text on CLICK with more information about the new Policy Manager grid
3. The filter control remains, but gains the default "All" option
   a. Other filter scope options are Type, Policy Name, and Notes
4. Create New policy now allows for group and property selection upon creation
5. Policy activation icons contain the version number of the policy active on the identified network
   a. Green = #50BE1F - used to indicate active status
   b. Yellow = #FFDD00 (black text inside) - to indicate "pending" status of a policy that is in the process of being activated
   c. Red = #B21701 - used to indicate a failure or problem
   d. A white icon with a dashed border means that this policy version is active on the property listed, but the property itself is not active on a network
6. Menu cog is now a Pulsar-compliant button
7. Pagination controls are at the top and bottom (pending decisions around "endless scroll" page)
8. The Policy list page should, by default, be sorted on the Policy Name, by alpha order.
9. The Cog menu button reveals this list of options



Policy Detail Page (Version list)

This is the main Policy Detail page

1. Shows the breadcrumb with the previous page clickable
   a. breadcrumb links gain underline on hover
2. The header bar of the Policy Details box shows the name of the policy as well as its cloudlet type
3. The Activate Policy button provides a convenient way for users to activate any version of this policy
4. Policy Details
   a. shows activity on different networks using the same color scheme as described on page 1
   b. Also shows policy notes
5. The Property details
   a. show the name of the property to which this policy is bound
      i. The binding property may be selected here to "reserve" a policy for a given property, if the binding has not been determined on the Property side.
   b. Shows the status and versions of that property on the networks
   c. The Group Association for this Policy is show here - policies can be moved or cloned to new groups from the main Policy List page
6. The Policy Detail page should, by default, be sorted by the Version number, in reverse order (latest version at the top).
   a. Version number cells act as do other hyperlinked text link cells:
   b. the entire cell should be interactable and clickable
   c. The text should gain an underline on interaction
7. A policy's status as locked now shows at the right edge of the Notes column.
   a. A policy is locked if it has ever been successfully activated on a network
   b. If the policy is locked, the lock appears
8. The network columns now contain more up-front information about each policy version
   a. Colors conform to the previously defined standards
   b. If a policy is inactive on a network, the column shows the date of the last activation
   c. If a policy has never been activated on a network, the column remains empty
9. The Actions column now presents all tools:
   a. Activate control
   b. Edit/view control
      i. While policy is unlocked, the edit Pencil is presented
      ii. When a policy is locked, that changes to the View icon
   c. Clone Version control
   d. Note that Version delete is not included in 15.3
10. Policy Notes are editable from the Policy Details page.
    a. This is a text area that auto-saves the user's input a predefined interval after the last edit, or on blur.
    b. The state is shown in italic text under the field label:
       i. "Editing": shown while the user is entering data, until the save operation begins.
       ii. "Saving...": the field is saving (if needed)
       iii. "Saved": the contents have been saved

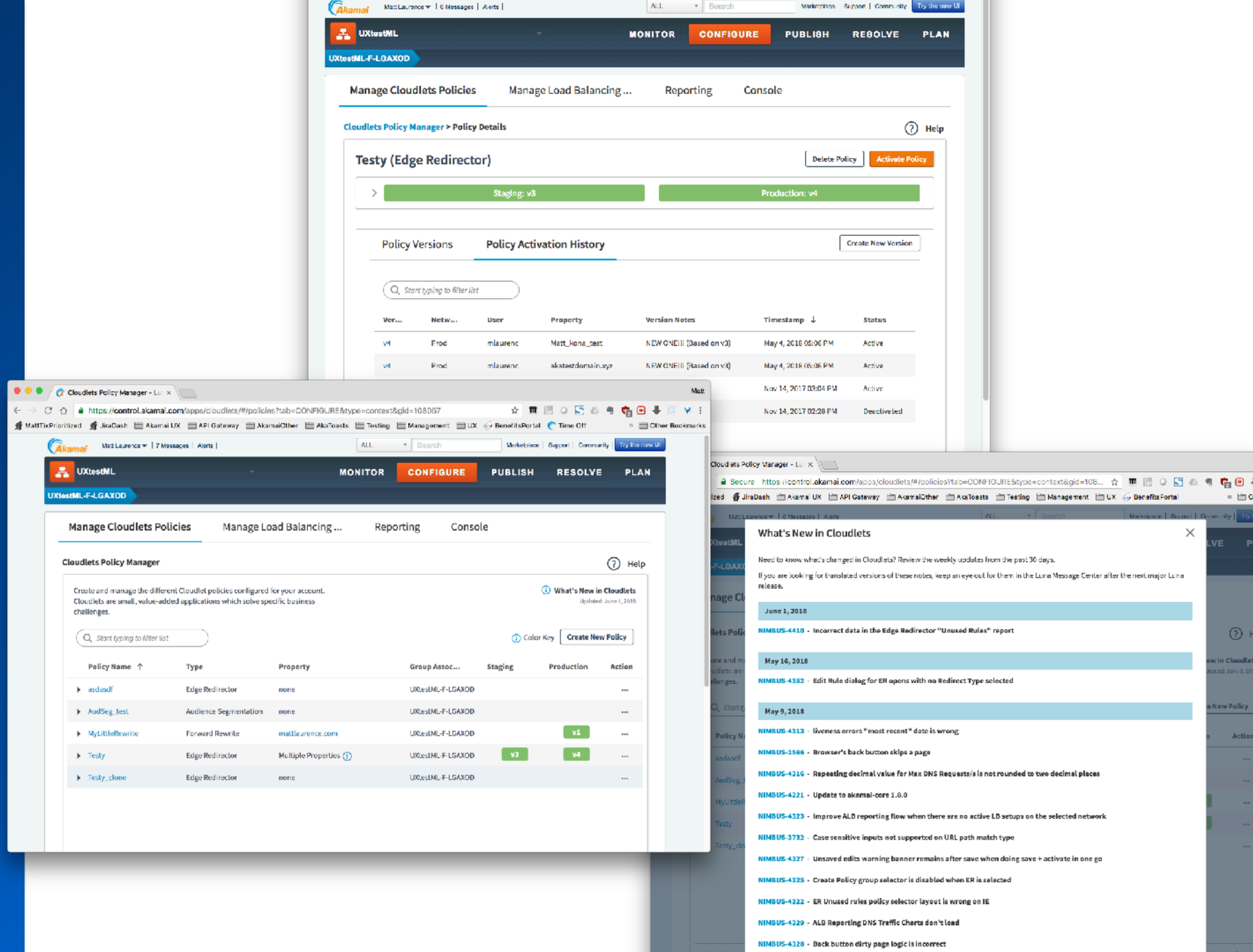


Phase 3: Create

Once the new UI was built, we did more testing to find out what else we needed to address. The prioritized results of those tests formed the backlog to be addressed by the team.

We simultaneously began building new functionality and rolling out new Cloudlets with different and more complex functionality.

Follow up surveys and usability testing showed strong increases in usability, approachability, and efficiency for the overall product.

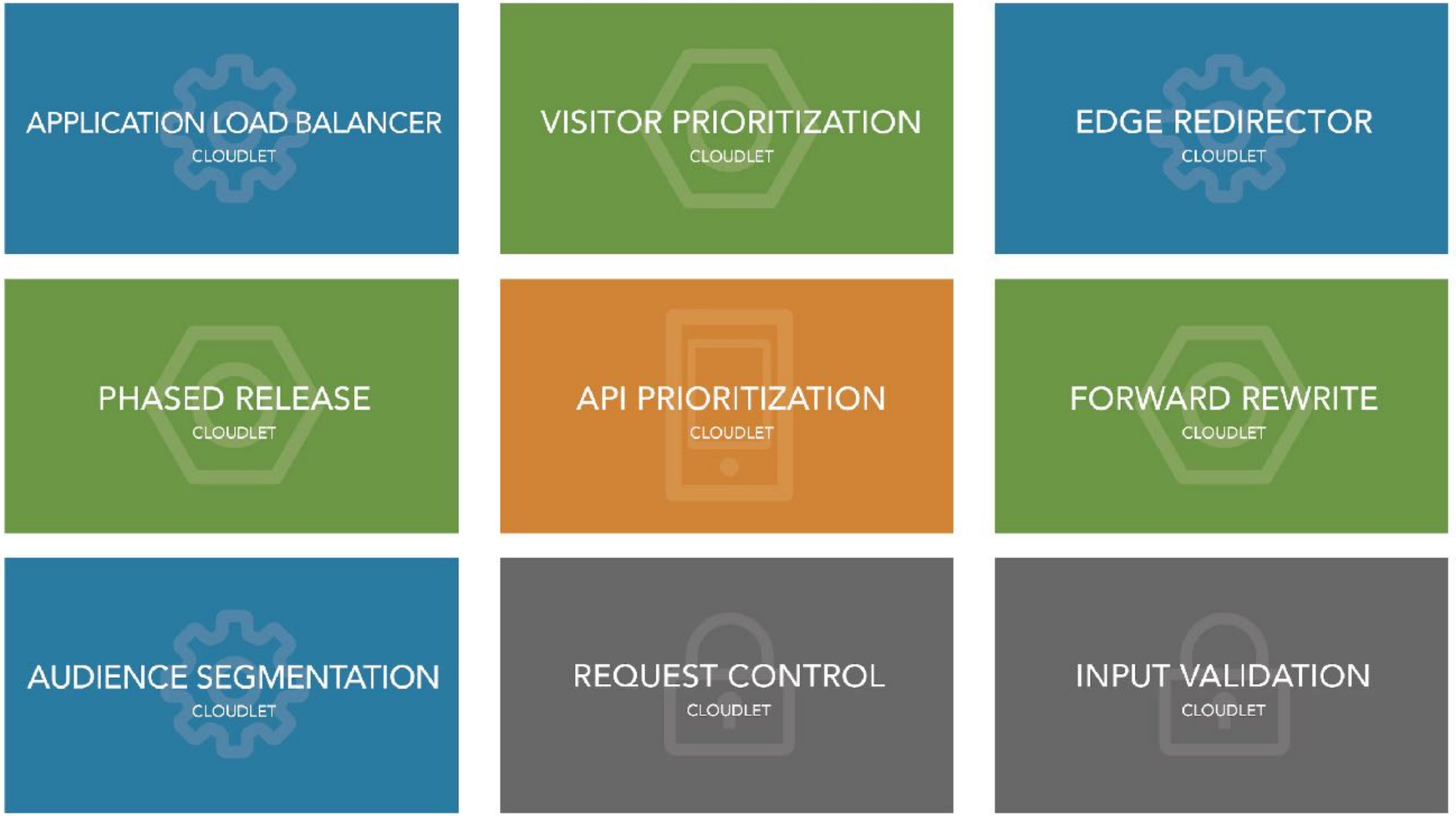


Results

As often happens at Akamai, I was moved to a higher priority project in mid-2016. But the Cloudlets team continues to come to me for insights during the UX Office Hours process that I started for unsupported products during that time.

There are now nine Cloudlets, each representing very different pieces of logic that can be moved to the Edge for notable performance increases.

It continues to be a cornerstone product for many of our largest customers.



Thank you!

Matt Laurence
mattlaurence@gmail.com