

What You Need to Know Before Launching Your API

Learnings from  **svix** and other APIs

Tom Hacoen
API World 2022

@TomHacoen
www.svix.com

Who am I?

- Founder and CEO of Svix - Webhooks as a Service
 - Help companies send webhooks
 - An API first developer tool
- Open source dev and maintainer
- Previously led teams at Samsung and the Israeli intelligence corps



What is this talk about?

- What to know before launching your API
- Learnings from Svix and other services
 - Your requirements may differ
 - Apply reasonable judgement
- Forgot to mention anything? Let me know!



General Guidelines



Keep It Simple Silly

- Complexity is your biggest enemy - avoid if possible
- Prevents you from moving fast
- Many points of failure
- Hard to reason about

There is good debt and bad debt

Good debt



Bad debt



Understand your tech stack

- Different technology comes with different trade-offs
- Do you care about consistency? Availability?
- Know the tools you use and their limitations



Understand the Problem

Know your customers


- What do they want?
- What do they care about?
- What would they hate?
- Have the curiosity of a child.



*“If I asked people what they wanted, they
would have asked for a faster horse.”*

— Henry Ford



A close-up, low-angle shot of a person's hand holding a black power drill. The drill is positioned vertically, with its bit just above a hole in a long, grey metal rail. The rail is resting on a dark, textured workbench. In the background, a red container filled with drill bits is visible, along with other tools like a wrench and a screwdriver. The scene is dimly lit, with the focus on the drill and the rail.

It's not what you do,
it's what you enable



Understand the solution

- Have a deep understanding of your chosen solution
- But be flexible and dynamic
- *“Everyone has a plan until they get punched in the mouth” — Mike Tyson*

An aerial view of a construction site on a steel deck. Two workers in orange safety gear and hard hats are visible. One worker is kneeling near a yellow power tool, while the other stands holding a green strap. The deck is covered with rebar cages, loose rebar, and various construction materials. The text "Set the right foundations" is overlaid in white.

Set the right foundations



Manage API complexity

- Strive for simplicity, and learn to say no
- Be explicit, watch out for accidental flexibility
 - Especially watch out with GraphQL
- Can't break API - if you build it, you're stuck with it
 - Try to plan for forward compatibility
- Don't leak implementation details

How do you deploy?

- SaaS? On prem? Both? Hybrid?
- Are you limited to AWS? GCP? Only PostgreSQL?
- Have any other runtime dependencies?

The background of the slide is a photograph of a large, modern library. It features multiple levels with white bookshelves filled with books. There are several blue armchairs and small tables for study or reading. The lighting is soft and even. The text is overlaid on this image.

Choose your data layer wisely

“Bad programmers worry about the code. Good programmers worry about data structures and their relationships.”

— Linus Torvalds

A dark brown goat with a rope around its neck, standing in a dry, dusty environment. The goat is facing left, and its body is covered in thick, dark fur. The background is a dry, dusty landscape with some sparse vegetation.

People will still use it wrong

- Make it easy to use right
- Make it hard to use wrong
- Expect it to be used wrong and be ready

Find your north star

- For us it's RELIABILITY, so:
 - Never lose a message once accepted
 - Avoid downtime at all costs
 - High speed and low latency

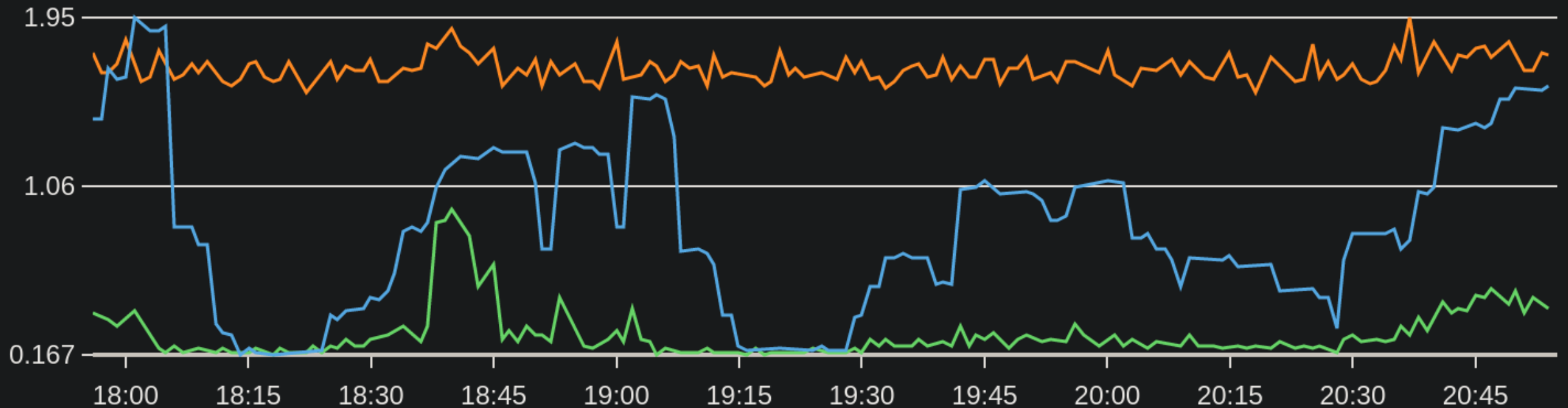
An aerial photograph of a large cable-stayed bridge spanning a wide body of water. The bridge features two prominent white pylons with multiple stay cables supporting the deck. The water is dark blue with some ripples, and a small boat is visible in the lower center. In the background, a cityscape and other land areas are visible under a hazy sky. The text "Be dependable" is overlaid in white, sans-serif font in the center of the image.

Be dependable

Monitor everything

Have fully visibility into your systems

Percent





WHERE'S
WALDO?

Be careful with noise and aggregations

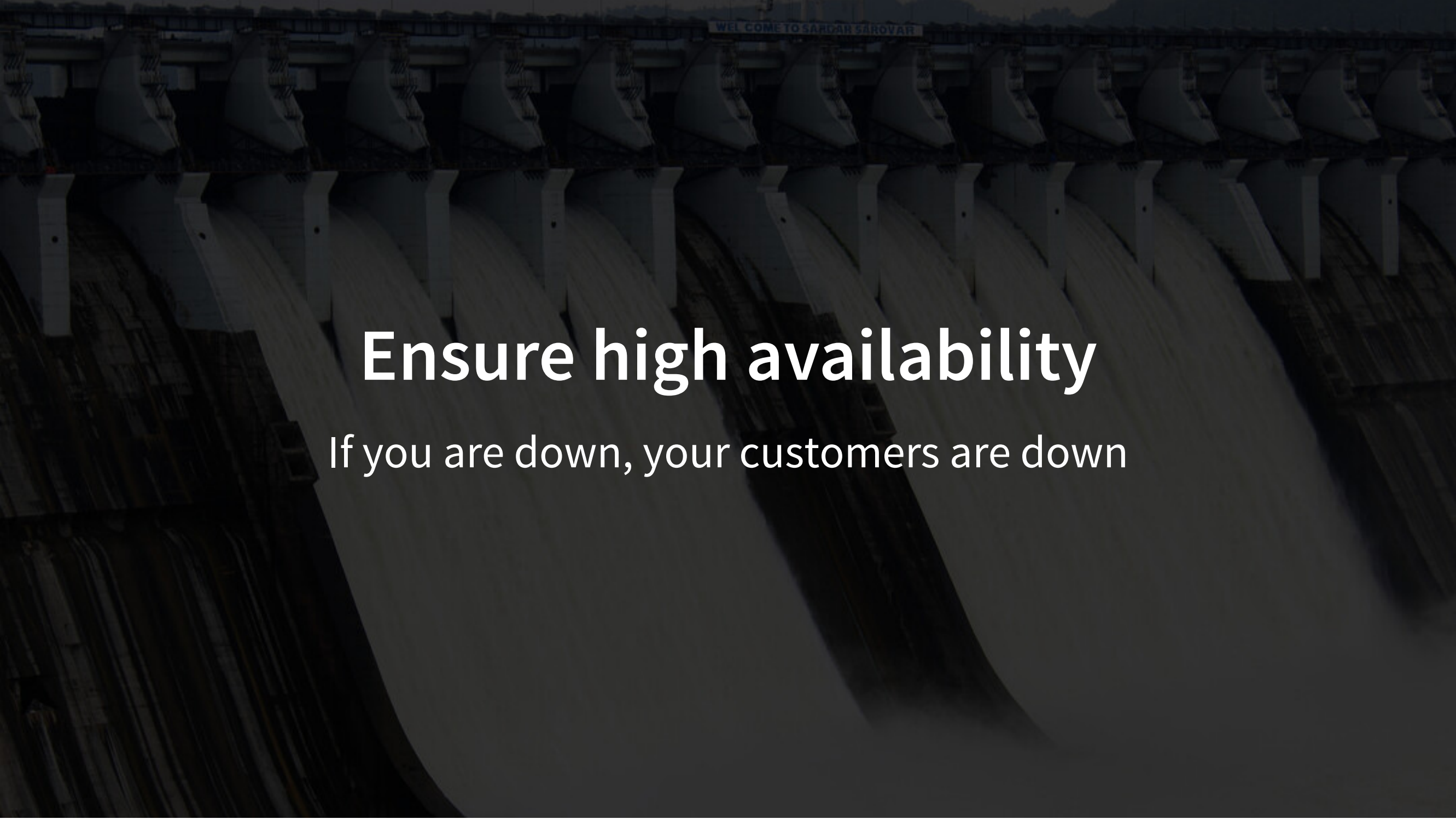
- Too noisy:
 - Easy to miss issues
 - Team will get used to ignoring alerts
- High-level aggregations and graphs are great
 - Easy to miss the details - have granularity

Logs everywhere

- You want to be able to detect anomalies
- Have information when something goes wrong
- Be able to trace the whole request
- Watch out not to log:
 - PII & PHI
 - Sensitive information

Reliability

- Have redundancies in place
- No single point of failure
- Test your systems under load
- Seek out failure cases and address



Ensure high availability

If you are down, your customers are down

Accountability

- Have a public status page
- Be quick and transparent about failures
- Reach out to affected parties immediately
- Do post-mortems to understand failures
- Hold yourself to the highest standards

Define and test behaviors

- Test all routes and behaviors
- Use regression tests to lock behavior
- Ensure high code coverage (though not enough!)
- Test changes and have ongoing tests
- We encode constraints in the type system —
thank you Rust!



Disaster recovery

- Hope for the best, prepare for the worst
- Have off-site backups and point-in-time recovery
- List what can go wrong and how to address
- Train staff on recovering from failures

Know your dependencies

- You're only as strong as your weakest link
- Only use trusted tools from trusted sources
- Make sure your tools are dependable
- Evaluate your dependencies yourself
- It's OK to be bold, if you are also prudent

Know the metrics your customers care about

- Uptime in terms of % per month?
- Uptime in terms of # of API errors?
- Do they care about latency? Throughput?
- Strong or eventual consistency?




Don't aim for 100% uptime

- 100% uptime is not achievable, you gotta stop somewhere
 - E.g. destruction of the earth is out of scope
- Law of diminishing returns
- Chasing 100% can make things more brittle
 - Which leads to less uptime...



Application security

- OWASP Top 10 and best practices
- Correct password handling
- Strict authorization and restrictions



Operational security

- Vulnerability handling - patching systems
- Security and dependency scanners
- Security training for the team
- Encryption in transit and at rest



Strict and locked down

- Least privilege access control
- No one should have direct access to prod
- Limit capabilities of accounts with access
- Paper trail - log all access and operations



Isolating data between customers

- Multi tenant? Single tenant?
- Can one customer access another's data?
- Have strong enforcement for that



Make security easy for your customers

- Role based access control
- Enable key rotation and scoped keys
- Educate them about security with your service
- Make it hard to get security wrong




Educate yourself with security best
practices




It's more than just code

- There is a lot of devops
- There is a lot of infrastructure
- Use infra-as-code - don't touch the UI!
- Build or buy? I prefer buy

A perspective view of a modern glass and metal walkway or bridge, receding into the distance. The walkway is flanked by glass railings and metal supports, creating a sense of depth and architectural structure. The background shows a dark, possibly industrial or urban setting, with some lights visible in the distance.

**Prepare for the future,
but build for the present**

A large yellow construction crane is the central focus, its lattice boom extending diagonally across the frame. The crane is set against a dark, overcast sky. In the background, other cranes are visible, including a blue one on the right and a smaller yellow one in the bottom left corner. The overall tone is industrial and somewhat somber due to the dark lighting.

Make it scalable, but not too much

- Build for 2-5x your current expected scale
- Have a path in mind for 10-20x

An aerial photograph of a dense city skyline, likely New York City, taken from a high vantage point. The image shows a vast number of skyscrapers and buildings, with the city extending to the water's edge. The sky is a deep blue with some light clouds, suggesting dusk or dawn. The overall tone is dark and atmospheric.

Multi region, multi environment

- Customers want a test environment
- Same region means lower latency and higher reliability
- Watch out for splitting your internal data across regions

Role based access control

- Your customers will want different users
- Your customers will want different roles
- Easier to keep it in mind from the start





Be flexible and extensible

- Your customers are developers
- They may want to use it differently
- Let them build cool things with your product

Send webhooks

- Enables real-time interactions with your system
- Your customers want it to build integrations
- Make sure it's reliable (retry, scaling, etc.)
- Watch out for security implications (SSRF, MITM, replay)
- Don't forget about monitoring, fanout, more...

The devtool equivalent of UX

A close-up photograph of a person's hand holding a black pen, drawing a wireframe on a notepad. The wireframe consists of several rectangular boxes connected by lines, representing a user interface layout. The hand is wearing a ring on the ring finger. The background is dark and out of focus.



Be consistent

API Standards vs. BDFL

Great docs make the difference

- Both overview and deep-dive
- Both beginners and advance
- Show how your API should be used
- Don't assume people know your product
 - The curse of knowledge



Don't forget about tutorials

- Guide developers through common uses
- Get them started quickly - easy onboarding
- Highlight cool features

Even developers need support

- Maybe your API isn't clear
- Maybe they found a bug or want a feature
- Maybe they just need help
- Make it easy to debug and report issues

Have a great API

Good

```
await fetch('https://api.svix.com/api/v1/app/', {  
  method: 'POST',  
  headers: {  
    'Content-Type': 'application/json'  
    'Accept': 'application/json'  
    'Authorization': 'Bearer ' + token,  
  },  
  body: '{"name": "some name", "uid": "app_234"}',  
});
```

Have an awesome SDK

Better

```
const svix = new Svix(token);  
  
await svix.application.create({  
  name: "some name",  
  uid: "app_234",  
});
```


A few SDK tips

- Consistent with language first, across SDKs second
- User-Agent: `svix-libs/1.29.0/python`
- X-Request-ID: `f058ebd...344e8cde5`
- Automatic retries in short intervals
 - Include attempt count in header
 - Reuse request-id
- Show X-Request-ID in SDK errors

A few API tips

- People want PATCH, in addition to PUT
- Support idempotency for POST requests
- Probably: pagination with iterators, not offsets
- List responses should be objects:

```
{  
  "data": [...],  
  "done": true,  
  "next": "app_1uikwVvIwNtmMXmwOnibro8nsH1",  
  "prev": "-app_1v2QtUrfQvWLUhBYf35n0JY4yIo",  
}
```


A knight in full plate armor is riding a white horse. The knight is holding a sword aloft in his right hand. The horse is wearing a surcoat with horizontal blue and yellow stripes. The background is a dark, out-of-focus landscape.

Be defensive and helpful

Tag your auth tokens

testsk_1F00EQKwBr7VYC0qpFW7XGYIBycWgqcB.eu

- test: optional test environment indicator
- sk: type of key (secret key, public key, etc.)
- 1F00EQKwB...BycWgqcB: random token
- eu: region

Add the shape to secret scanning databases

Tag your IDs

```
// Application token  
app_29we3mZemNijHrQcrLlJG1pRCst
```

```
// Endpoint token  
ep_1uikje8Xw8Z3GaSwtUYmIBhhYTN
```

```
// Message token  
msg_270EWwmNfwpCwutpZ4GVeypLvvp
```

Support user-defined IDs

```
await svix.application.create({  
  name: "some name",  
  uid: "my-app-123", // This is your customers' internal ID  
});  
// Created ID: app_29we3mZemNijHrQcrLlJG1pRCst  
  
// These IDs can then use interchangeably  
await svix.endpoint.list("app_29we3mZemNijHrQcrLlJG1pRCst");  
await svix.endpoint.list("my-app-123");
```

Custom metadata on entities

```
await svix.application.create({  
  name: "some name",  
  uid: "app_234",  
  metadata: {  
    "org": "organization 1",  
  },  
});
```




BONUS: Would like to offer on-prem?

- It's A LOT of additional work
- Support multiple environments and way to deploy
- You need more configuration options
- Even more documentation
- You have to think about data migrations
 - You can no longer manage that



BONUS: Billing an API product

- 80/20 revenue rule for API companies
- Experiment and find what works for you
- Usage based billing is common, SaaS billing helps predictability
 - Tie it to perceived value metrics
 - Make sure customers pre-pay credits
 - Display usage in real-time: avoid surprise bills

Closing words

- Main takeaway: it's not all about the code
- These worked for us, you may be different
- Thanks for their help:
 - Greg from Monday.com
 - Adam from Kable
 - Raffi and Anh-Tho from Lago
 - Aleks from WordCab

Questions?

- For webhooks, check out  **svix** at www.svix.com
- Something missing? Tweet at [@TomHacohen](https://twitter.com/TomHacohen)