С чего начать

Предиктека — установка, которая держит справочники, печатает документы и выполняет долгие задачи для одного или нескольких workspace. Всё работает на вашем сервере: ни данные, ни документы не уходят наружу, и для печати не нужен браузер.

Три шага

1 · Войдите. Адрес кабинета и первый администратор выдаются при установке; пароль меняется в профиле. Один вход действует на все workspace, где вы участник.

2 · Выберите workspace. Workspace — это рабочая среда организации: свои участники, свой тариф, свои данные. Переключение между ними — в шапке, права проверяются на каждом запросе.

3 · Загрузите первый файл. Страница «Задачи» принимает CSV со списком контрагентов, товаров или видов деятельности, а затем запускает сопоставление со справочником. Результат — файл, который можно скачать, и очередь строк, где система не уверена: их подтверждает человек.

Что дальше

- Документы и шаблоны — как напечатать договор и пакет договоров.
- Подключение ИИ — как дать ассистенту доступ к справочникам и печати.
- Тарифы — что входит в тариф и как поставить лицензию на свой сервер.

Если что-то не работает

Ошибки в кабинете пишут, что именно поправить: «ИНН покупателя: обязательное поле отсутствует» вместо пустой строки в договоре, «Тип файла не поддерживается» вместо зависшей загрузки. Если сообщения не хватает, откройте «Журнал» — там видно, какое действие выполнялось и с каким объектом.

Документы и шаблоны

Шаблон — это **схема полей** и **рисунок страницы**. Схема описывает, какие данные нужны документу; рисунок определяет, как они выглядят. Данные проверяются по схеме **до** печати, поэтому пустых строк в договоре не бывает.

Два движка

Движок	Что это	Когда выбирать
`typst`	вёрстка на Typst, печать в PDF, PNG и SVG	нужен аккуратный документ, который делает система
`docx`	настоящий документ Word с плейсхолдерами	у компании есть свой бланк, и договор правят в Word

Word-шаблон — это файл `.docx`, который вы открываете в Word и правите как обычно: реквизиты, шрифты, колонтитулы, логотип. Плейсхолдеры остаются на месте, и при печати подставляются данные.

Как установить шаблон

1 · Соберите zip: `manifest.json`, `schema.json` и файл шаблона (`main.typ` или `template.docx`).

2 · Откройте «Документы» → «Загрузить шаблон» и выберите архив.

3 · Шаблон проверится: схема компилируется, Typst собирается, Word-файл открывается. Если что-то

не так, вы узнаете об этом сразу, а не в момент подписания.

Версия шаблона неизменяема: правка — это новая версия, а каждый напечатанный документ помнит, по какой версии он сделан, и хранит контрольную сумму файла. Через год видно, что именно печатали.

Печать одного документа

Откройте шаблон, заполните поля (форма показывает пример из схемы) и нажмите «Напечатать». Если нужны приложения, отметьте файлы из папки workspace: они будут переведены в PDF и сшиты с документом в один файл — договор первым, приложения за ним, в порядке отметки.

Что поддерживается как приложение

PDF берётся как есть. Всё остальное переводится в PDF встроенным движком: `.docx`, `.md`, `.html`, `.csv`, `.txt`, изображения. Сканы лучше прикладывать PDF-ом: так сохраняется разрешение.

Если приложение не удалось включить — файл не читается, формат не переводится, путь ведёт вне workspace — **документ не печатается**. Ответ называет файл и причину. Это осознанный выбор: договор, отданный на подпись без спецификации, хуже отказа, потому что пропажа обнаруживается позже. В пакете такая строка попадает в «не напечатано» с той же причиной.

Плейсхолдеры

Синтаксис `{{путь}}` — это **движок Word**. Шаблон Typst плейсхолдеров не знает: он читает тот же `data.json` и обращается к полям как к переменным. Два движка, два способа подставить данные, и путать их — самая частая ошибка при первом шаблоне.

```
{{buyer.name}}             значение поля
{{items.0.qty}}            первый элемент массива
{{#each items}}…{{/each}}  повтор строки таблицы для каждого элемента
{{this}}                   сам элемент внутри повтора
```

Путь — имя поля из схемы; точки разделяют вложенность, числа обозначают элемент массива.

В Word

Плейсхолдер можно набрать в любой ячейке таблицы или в абзаце. Word часто разрывает текст на несколько кусков форматирования; заполнение это переживает — куски склеиваются перед подстановкой.

Повтор строки ставится **внутри** строки таблицы: `{{#each items}}` в первой ячейке, `{{/each}}` в последней. Заполнение повторит всю строку столько раз, сколько элементов в массиве.

Значения экранируются: «Ромашка & Ко» попадёт в документ как есть и не сломает файл. Если поля в данных нет, подставится пусто — а обязательные поля схема не пропустит заранее.

В Typst

Плейсхолдеров нет. Данные приходят файлом `data.json` рядом с шаблоном, и шаблон читает их как обычно:

```
typst
#let d = json("data.json")
#let opt(dict, key) = if key in dict { dict.at(key) } else { "" }

Поставщик: #d.seller.name
Город: #opt(d, "city")
#for item in d.items [• #item.name — #item.qty шт. ]
```

Необязательное поле читайте через `opt`: прямое обращение к отсутствующему ключу — ошибка печати, и это правильно, но неудобно.

Схема

Схема — обычный JSON Schema (draft 2020-12). Обязательные поля перечисляются в `required`, а `example` показывается в форме как подсказка.

```
json
{
"type": "object",
"required": ["number", "date", "buyer"],
"properties": {
"number": {"type": "string", "example": "14/2026"},
"date": {"type": "string", "format": "date", "example": "2026-11-14"},
"buyer": {"$ref": "#/$defs/party"}
},
"$defs": {
"party": {"type": "object", "required": ["name", "inn"],
"properties": {"name": {"type": "string"}, "inn": {"type": "string"}}}
}
}
```

Сгенерировать шаблон без вёрстки

Если шаблона нет, его можно получить из списка полей — командой в кабинете или через ассистента (инструмент `scaffold_template`). Результат — готовый пакет: `typst` даёт аккуратную вёрстку, `docx` — документ Word, который компания правит под свой бланк.

Пакет документов

Пакет — это таблица, где одна строка даёт один документ. Так печатают двести договоров по списку контрагентов или сто актов по списку работ.

Как это устроено

1 · Загрузите CSV в папку workspace («Задачи» → «Загрузить файл»).

2 · На странице «Документы» выберите шаблон, файл и, если нужно, приложения.

3 · Нажмите «Напечатать пакет». Печать идёт прогоном: он переживает перезапуск сервера, а прогресс виден на странице «Задачи».

Заголовки колонок

Заголовок колонки — это путь поля из схемы, поэтому вложенные поля и позиции описываются точками:

```
csv
number,date,seller.name,seller.inn,buyer.name,buyer.inn,items.0.name,items.0.qty,items.0.price
1/2026,2026-11-14,ООО «Ромашка»,7707083893,ООО «Акме»,7707083894,Хлеб,2,89
2/2026,2026-11-14,ООО «Ромашка»,7707083893,ООО «Лютик»,7707083895,Молоко,1,74.5
```

Разделитель — запятая или точка с запятой (русский Excel пишет второе). Текстовые ячейки приводятся к типам схемы: `178` становится числом, `2026-11-14` — датой. Ничего не нужно оборачивать в кавычки вручную.

Ошибки в строках

Одна плохая строка не останавливает пакет. Она попадает в счётчик «не напечатано», а причина называется в предупреждениях: «строка 37: ИНН покупателя: контрольная сумма не сходится». Остальные документы печатаются.

Приложение, которое не удалось включить, — тоже провал строки, а не заметка: договор без спецификации в архиве неотличим от полного, и обнаружится это на подписании.

Результат

Все документы складываются в архив: `batch-<прогон>.zip`. Если выбраны приложения, каждый документ пакета получает их сшитыми в один PDF — так же, как при печати одного документа.

Подключение ИИ

Ассистент работает с Предиктекой по протоколу MCP: он видит те же справочники, шаблоны и прогоны, что и человек в кабинете, и только то, что разрешено тарифом и ролью.

Как подключить

1 · Откройте «Подключение ИИ» в workspace и нажмите «Выпустить новый токен».

2 · Скопируйте блок конфигурации: адрес и токен уже подставлены.

3 · Вставьте блок в настройки ассистента (ChatGPT, Claude, Cursor, VS Code — любой клиент, который умеет MCP по HTTP).

Workspace определяет **ключ**, а не адрес. Параметр `?workspace=` в адресе нужен только для читаемости: если он называет другой workspace, сервер отвечает 403, а не пишет документы в тот, которому принадлежит ключ. Поэтому адрес и ключ берутся из одного блока.

Токен показывается один раз и хранится как хеш. Если он потерян, выпустите новый: старый можно отозвать.

Что умеет ассистент

Инструмент	Зачем
`list_document_templates`	какие шаблоны есть в workspace
`get_template_schema`	какие поля нужны шаблону — точная схема
`render_document`	напечатать документ, проверив данные по схеме
`scaffold_template`	собрать пакет шаблона из списка полей (Typst или Word)
`start_batch_documents`	запустить печать пакета по CSV
`get_usage`	расход моделей и счёт

Инструментов ровно шесть, и это весь доступ ассистента к продукту: справочников, проверки ИНН и сумм прописью в MCP-блоке нет — они живут в сервисе данных и в API, но не здесь. Обещать ассистенту больше, чем блок умеет, значит получить от него уверенный ответ о том, чего он не проверял.

Почему это безопасно

- Ключ привязан к workspace: чужой workspace ассистент не увидит, даже зная адрес.
- Данные не покидают установку: модель получает только то, что человек или ассистент передал в запросе.
- Каждое действие ассистента попадает в журнал: кто, что, когда.

Модели

Если подключён роутер моделей, тот же ключ открывает доступ к моделям: ассистент не хранит ключи провайдеров, а расход считается по workspace и виден на странице «Расход».

HTTP API

Всё, что делает кабинет, можно сделать запросом: интерфейс — это клиент того же API. Ответы всегда в одном конверте: `data` и `meta`, а ошибки — `{"error": {"code", "message"}}` с подходящим кодом.

Адреса

Метод	Путь	Что делает

`GET` | `/api/me` | кто я и в каких workspace состою | `GET` | `/api/workspaces` | список workspace | `GET` | `/api/plans` | тарифы: хостинг и свой сервер | `POST` | `/api/runs` | запустить прогон: `{workspace, kind, input}` | `GET` | `/api/runs/{id}` | состояние прогона | `POST` | `/api/runs/{id}/review` | решение по спорным строкам | `GET` | `/api/mcp-config?workspace=` | готовая конфигурация MCP | `POST` | `/api/license` | поставить лицензию | `GET` | `/api/license` | какая лицензия стоит и до какого числа | `GET` | `/api/analytics` | счётчики отправки событий: отправлено, отброшено, неудачно | `GET` | `/api/usage?workspace=` | расход моделей и счёт | `POST` | `/mcp` | MCP-сервер (нужен токен workspace) | `GET` | `/docs`, `/docs.pdf` | документация страницами и одним файлом | `GET` | `/logo.svg`, `/favicon.svg` | знак продукта |

Пример: запустить сопоставление

```
bash
curl -X POST https://predicteka.example/api/runs \
-H "Cookie: pk_session=…" -H "Content-Type: application/json" \
-d '{"workspace":"acme","kind":"import",
"input":{"source":"inbox/contractors.csv","classifier":"okved2"}}'
```

Ответ `202 Accepted` содержит `id` прогона. Дальше `GET /api/runs/{id}` показывает статус, а `POST /api/runs/{id}/review` принимает решение по строкам, где система не уверена.

Идемпотентность

Идентификатор прогона — производная от работы: workspace, вид, файл и то, что меняет результат (классификатор для сопоставления, шаблон для пакета). Повторный запрос с теми же параметрами не создаёт второй прогон, а возвращает существующий — это спасает от двойной отправки, когда интеграция повторяет запрос после таймаута. Тот же файл с другим шаблоном — другая работа, и это другой прогон.

Ограничения

Файл — до 32 МБ. Пакет документов — до 200 строк за прогон (печатайте частями). Тяжёлые операции (печать, сшивка, модели) идут через очередь, поэтому ответ приходит сразу, а работа продолжается.

Тарифы и лицензия

Тариф определяет, что доступно в workspace: прогоны, документы, исследования, своя модель авторизации, роутинг моделей. Тарифы бывают двух видов: тот, что работает у нас, и тот, что вы ставите на своём сервере.

Как оплатить

Тариф покупается картой прямо в кабинете: «Тарифы» → «Перейти». Оплата идёт на странице провайдера, кабинет её не видит и данных карты не получает. Период считается от даты окончания текущего, продление заранее ничего не съедает. Подробности — Оплата и подписка.

Купить или продлить может владелец и администратор; участник видит цену и дату окончания.

Свой сервер

Лицензия — подписанный файл. Установка проверяет подпись публичным ключом и **не обращается к нам**: поэтому она работает в закрытом контуре и не зависит от доступности нашего сервиса.

Чтобы поставить лицензию: «Тарифы» → вставьте файл → «Установить и активировать». Активация идёт прогоном: он проверит подпись, запишет файл в папку лицензий и будет следить за сроком. Продление — новый файл; прежний остаётся в папке для аудита.

Что нельзя подделать

- Подпись. Правка файла ломает подпись, поэтому «дописать себе тариф» нельзя.
- Workspace. Лицензия выдана на конкретный workspace: копия на вторую установку не работает.
- Возможности. Список возможностей внутри подписанного файла, а не выводится из названия тарифа.

Отзыв

Установка раз в сутки забирает список отозванных лицензий и хранит последнюю копию. Если сети нет, она продолжает работать: у клиента с упавшим каналом данные важнее формальности. Отзыв действует с первого успешного опроса.

Расход моделей

Если подключён роутинг моделей, в тариф входит объём токенов, а сверх него считается по прайсу. Расход видно на странице «Расход», файл для финансов — там же, кнопкой «Скачать расход в CSV».

Аналитика продукта

Предиктека считает, чем её пользуются, и отправляет эти события на **свой** хост Porcupine — самостоятельный сервис продуктовой аналитики: принимает события, кладёт их в SQLite и проецирует в DuckLake, где к ним пускают SQL.

Внешних сервисов нет: адрес задаёт оператор установки, а если он не задан — не отправляется ничего. Установка в закрытом контуре работает ровно так же, только без событий.

Что отправляется

Событие	Когда	Свойства

`signed_in` | успешный вход | способ входа | `template_installed` | установлен шаблон | шаблон, версия, движок, форматы, число полей | `document_printed` | напечатан документ | шаблон, версия, движок,

формат, приложений, страниц, байт | `document_refused` | документ отклонён до печати | шаблон, причина (данные не по схеме, приложение не включилось) | `batch_started` | запущен пакет | шаблон, строк, приложений | `import_started` | запущено сопоставление | классификатор | `licence_installed` | поставлена лицензия | откуда | `mcp_token_issued` | выпущен ключ ассистента | — | `mcp_tool_called` | вызов инструмента ассистентом | инструмент, успех, миллисекунды, ошибка | К каждому событию добавляются `workspace`, `plan`, `service` и `build`.

Ассистент считается так же

Каждый вызов MCP-инструмента — событие `mcp_tool_called`: какой инструмент, сработал ли, сколько занял и текст ошибки, если не сработал. Это самая полезная часть аналитики, потому что ассистент — самый активный пользователь продукта, а его отказ («шаблона нет», «данные не по схеме») — это готовый список того, что людям мешает.

Пишется вызов, а не его содержимое: имя инструмента, workspace, тариф, длительность. Данные документа — тайна клиента; то, что ассистент напечатал договор, — метрика. По той же причине журнал действий не хранит тела документов.

Актор — хеш токена: один ассистент считается одним актором, два разных различаются, сам токен никуда не уходит.

Чего не отправляется

Содержимого документов, имён файлов, адресов почты, названий контор и workspace. Человек превращается в псевдоним — хеш его идентификатора записи (`u_9f2c…`), workspace — в свой случайный идентификатор. Печатать договор — рабочая тайна; то, что его напечатали, — метрика.

Где смотреть

Кабинет: страница **`/analytics`** — доступна владельцу workspace, потому что показывает цифры всей установки. Там же таблица событий, поле SQL и четыре готовых вопроса: печати по дням, какие шаблоны, вызовы ассистента, ошибки ассистента. Результат запроса — таблица, столбцы в том порядке, в котором сервер их вернул.

Чтение настраивается отдельно от записи: ключ записи умеет только добавлять события, а странице нужен сервисный ключ с правом `events:read`.

```
bash
export PORCUPINE_PROJECT=d3346824-…              # id проекта
export PORCUPINE_API_TOKEN=pc_sk_…               # сервисный ключ, право events:read
```

Кнопка **«Проверить связь»** отправляет пустой пакет и отвечает на первый вопрос оператора: неверный ключ или недоступный хост. Счётчики те же, что отдаёт `GET /api/analytics` в JSON — этого достаточно для мониторинга.

Как включить

```
bash
export PORCUPINE_URL=http://analytics.example.com
export PORCUPINE_WRITE_KEY=pk_...        # публичный ключ записи проекта
```

Перезапустите кабинет. Проверить, что связь есть: `GET /api/analytics` отвечает счётчиками отправленных, отброшенных и неудачных событий и последней ошибкой — по ним видно разницу между «никто не пользуется», «не настроено» и «хост отказывает».

Почему это не может сломать печать

Клиент `internal/analytics` отправляет события **в фоне**: у него ограниченная очередь, и при переполнении событие отбрасывается и считается, а не ждёт. Ошибка аналитики не влияет на печать документа, установку шаблона или запуск прогона — это проверено тестом `TestUnreachableHostNeverBlocks`.

Неудачи делятся на два вида: `429` и `5xx` повторяются с тем же `message_id` (дедупликация на стороне Porcupine, поэтому повтор не превращается в двойной счёт), а `4xx` не повторяется вовсе — неправильное событие не станет правильным от второй попытки.

Запросы

События лежат в таблице `events` проекта. Число напечатанных документов по дням:

```
sql
SELECT CAST(event_time AS DATE) AS day, count(*) AS printed
FROM events
WHERE event = 'document_printed'
GROUP BY day ORDER BY day
```

Какие шаблоны печатают, по убыванию:

```
sql
SELECT json_extract_string(properties, '$.template') AS template,
count(*) AS prints,
sum(CAST(json_extract_string(properties, '$.pages') AS INTEGER)) AS pages
FROM events
WHERE event = 'document_printed'
GROUP BY template ORDER BY prints DESC
```

Доля документов, напечатанных вместе с приложениями:

```
sql
SELECT count(*) FILTER (WHERE CAST(json_extract_string(properties, '$.appendices') AS INTEGER) >
0) * 100.0
/ count(*) AS with_appendix_percent
FROM events
WHERE event = 'document_printed'
```

Воронка: вошли → установили шаблон → напечатали:

```sql
SELECT
count(DISTINCT person_id) FILTER (WHERE event = 'signed_in') AS signed_in,
count(DISTINCT person_id) FILTER (WHERE event = 'template_installed') AS installed,
count(DISTINCT person_id) FILTER (WHERE event = 'document_printed') AS printed
FROM events
WHERE event_time > now() - INTERVAL 30 DAY
```

Свой экземпляр для разработки

```bash
git clone https://github.com/olup/porcupine && cd porcupine
# README → quickstart: создаёт .env со случайным bootstrap-секретом
docker compose -f compose.yaml -f compose.oauth.yaml -f compose.release.yaml up -d
curl --fail --silent --retry 30 --retry-all-errors --retry-delay 1 http://localhost:8080/readyz
```

Экземпляр поднимается на `http://localhost:8080`. Проект и ключ записи выдаются в его интерфейсе; дальше `PORCUPINE_URL` и `PORCUPINE_WRITE_KEY` выше — и события из кабинета поедут туда.

У Porcupine нет своих экранов: это API, MCP и CLI. Поэтому админка живёт в кабинете (страница `/analytics`), а не в отдельном приложении.

Внутренний домен

Если установка опубликована в tailnet, Porcupine удобно держать там же — наружу он не смотрит:

```bash
tailscale serve --bg --https=8443 18080
tailscale serve status
# https://<host>.<tailnet>.ts.net:8443 -> http://127.0.0.1:18080
```

Порт 8443, а не 443: 443 на сервере обычно занят обратным прокси, и `tailscale serve` в этом случае падает с `bind: address already in use`. `PORCUPINE_OAUTH_ISSUER` должен совпадать с этим адресом символ в символ, иначе OAuth-клиенты получат токен не для того ресурса.

Оплата и подписка

Тариф покупается картой на странице платёжного провайдера. Предиктека не видит данных карты и поэтому не попадает в область PCI DSS: браузер уходит на страницу провайдера, а к нам возвращается только состояние заказа.

Что происходит при покупке

1 · Владелец или администратор нажимает «Перейти» у нужного тарифа.

2 · Кабинет **сначала записывает заказ** — номер, тариф, сумму — и только потом идёт к провайдеру.

Поэтому уведомление, пришедшее на неизвестный номер, не может поднять тариф: такого заказа здесь нет.

3 · Человек платит на странице провайдера и возвращается в кабинет.

4 · Провайдер присылает уведомление. Оно подписано ключом установки, проверяется по телу запроса и

зачитывается ровно один раз.

Период считается от даты окончания текущего, а не от сегодняшнего дня: продление заранее не съедает уже оплаченные дни.

Кто платит

Купить или продлить тариф может владелец и администратор workspace. Участник видит тариф, цену и дату окончания, но кнопки оплаты у него нет: он не отвечает за счёт.

Что приходит от провайдера

Уведомление (`IPN`) — это JSON с подписью в заголовках. Предиктека отвечает `200` на всё, что поняла, включая повтор: провайдер повторяет отправку до пятидесяти раз в течение семи дней, и `200` на повтор — это способ остановить очередь, а не признание, что что-то изменилось.

Отказы:

Ответ	Когда

`401` | подпись не сходится или ключ принадлежит другому продавцу | `400` | тело не разобрано, нет номера заказа, либо сумма не совпадает с заказом | `503` | на установке не настроена оплата — обещать «принято» нельзя | `200` | всё понято: оплачено, отклонено, возвращено, повтор, или заказ не наш |

Сумма проверяется по заказу. Уведомление, которое зачитывает месяц за один рубль, — это либо ошибка, либо попытка, и без сверки они выглядят одинаково.

Возврат

На странице «Тарифы» у оплаченного заказа есть «Вернуть». Запрос уходит провайдеру, заказ помечается возвращённым, событие пишется в журнал.

Тариф при возврате **не снимается автоматически**: человек мог уже напечатать документы. Если нужно закрыть доступ, это отдельное, видимое действие. Возврат закрывает заказ: даже если после него придёт запоздавшее «оплачено», второго месяца за те же деньги не будет.

Когда период заканчивается

Период заканчивается в дату, которую никто не караулит. Раз в шесть часов прогон проверяет workspace с датой окончания и переводит просроченные на бесплатный тариф:

- документы, справочники и участники остаются;
- шаблоны, исследования и лишние участники становятся недоступны;
- в кабинете за неделю до конца появляется предупреждение.

Прогон живёт в Temporal (`ExpireSubscriptions`), а не в кабинете: срок истекает и тогда, когда никто не открывает страницы. Он запускается один раз на установку и переживает перезапуск воркера.

Настройка

```
bash
export YPMN_BASE_URL=https://sandbox.ypmn.ru/api   # боевой: https://secure.ypmn.ru/api
export YPMN_MERCHANT=MERCHANT_CODE                   # «Код продавца» в кабинете провайдера
export YPMN_SECRET=...                               # «Секретный ключ», только в окружении
```

Без `YPMN_MERCHANT` и `YPMN_SECRET` установка работает, но кнопок оплаты нет: страница говорит, чего не хватает, вместо кнопки, которая падает.

Два секрета установки: `YPMN_SECRET` подписывает запросы и проверяет уведомления, `HUB_SECRET` подписывает сессии. Оба живут в окружении, не в репозитории и не в образе.

Адрес для уведомлений

В кабинете провайдера регистрируется адрес установки:

```
https://<адрес установки>/payments/ipn
```

Адрес публичный — провайдер не держит сессию — и защищён подписью, а не списком адресов. Если установка стоит за прокси, который переписывает путь, подпись всё равно сходится: проверяются и путь запроса, и `X-Forwarded-Uri`.

Налоги

Ставка НДС уходит в заказе построчно; по умолчанию 22%. Это значение — для бухгалтера клиента, а не для кода: если у продавца другая ставка или другая страна, она задаётся в позиции заказа.