



# От апробации к системной цели

ГОД С ASTRA AUTOMATION

Как инженерная команда изменила  
модель управления инфраструктурой

6 дней → 18 минут

путь от ручной заявки к каталогу самообслуживания

Руководитель практики управления ИТ-мощностями /  
директор вычислительных ИТ-платформ  
Кирилл Носков

# Этапы проекта



Не презентация продукта, а история выбора, риска и масштабирования

01 Выбор

02 Пилот

03 Масштаб

04 Управляемость

1 500

серверов =

1 500

потенциально разных  
способов получить инцидент

Поэтому проект начинался не с автоматизации.

Он начинался с вопроса, как сделать  
инфраструктуру управляемой?



# Быстрый рост инфраструктуры



Начало 2024: гетерогенная среда, ручные согласования и уникальные конфигурации

**~1 500**

серверов: Astra Linux,  
CentOS, Windows

**6 дней**

среднее время заявки  
на стандартную VM

**8 часов**

восстановление сервера  
после сбоя

**60%**

времени администраторов  
уходило на рутину

Заявка



5 согласований



4 ручных этапа



Сервер

ручная вариативность

## Управленческий диагноз

Мы росли быстрее, чем могла масштабироваться ручная модель управления. Каждая уникальная конфигурация становилась потенциальным инцидентом.



# Пять решений, которые сделали масштабирование возможным



## Признать системную проблему

не ускорять хаос, а менять модель управления

01



## Сначала стандарт

эталонные шаблоны, единые политики, проверенный контент

02

## Выбирать по ROI

время, compliance, FTE, а не «нравится / не нравится»

03

## Пилотировать на живой среде

стенд не “сопротивляется”, настоящие ограничения видны в эксплуатации

04

## Не сокращать FTE

снять людей с рутины и направить на автоматизацию

05

## Новая роль технологии в докладе

Не «мы внедрили платформу», а «мы превратили технологический выбор в управляемую модель: критерии → пилот → стандарт → масштабирование → измеримый эффект»

# Риски: что нас пугало



Команда разложила риски на две группы: бездействие и внедрение

## Если ничего не делать

регуляторные требования ужесточаются

compliance-узлы стагнируют

уходящие инструменты без поддержки

рост инфраструктуры на 20-30% при неизменном штате

## Если внедрять

«еще одна платформа», которой не пользуются

сопротивление команды: «мы и так скриптами делаем»

ошибочная конфигурация размножается автоматически

зависимость от одного вендора

Главный риск

Автоматизация ошибочной конфигурации =  
ошибка, размноженная в 1 500 экземплярах



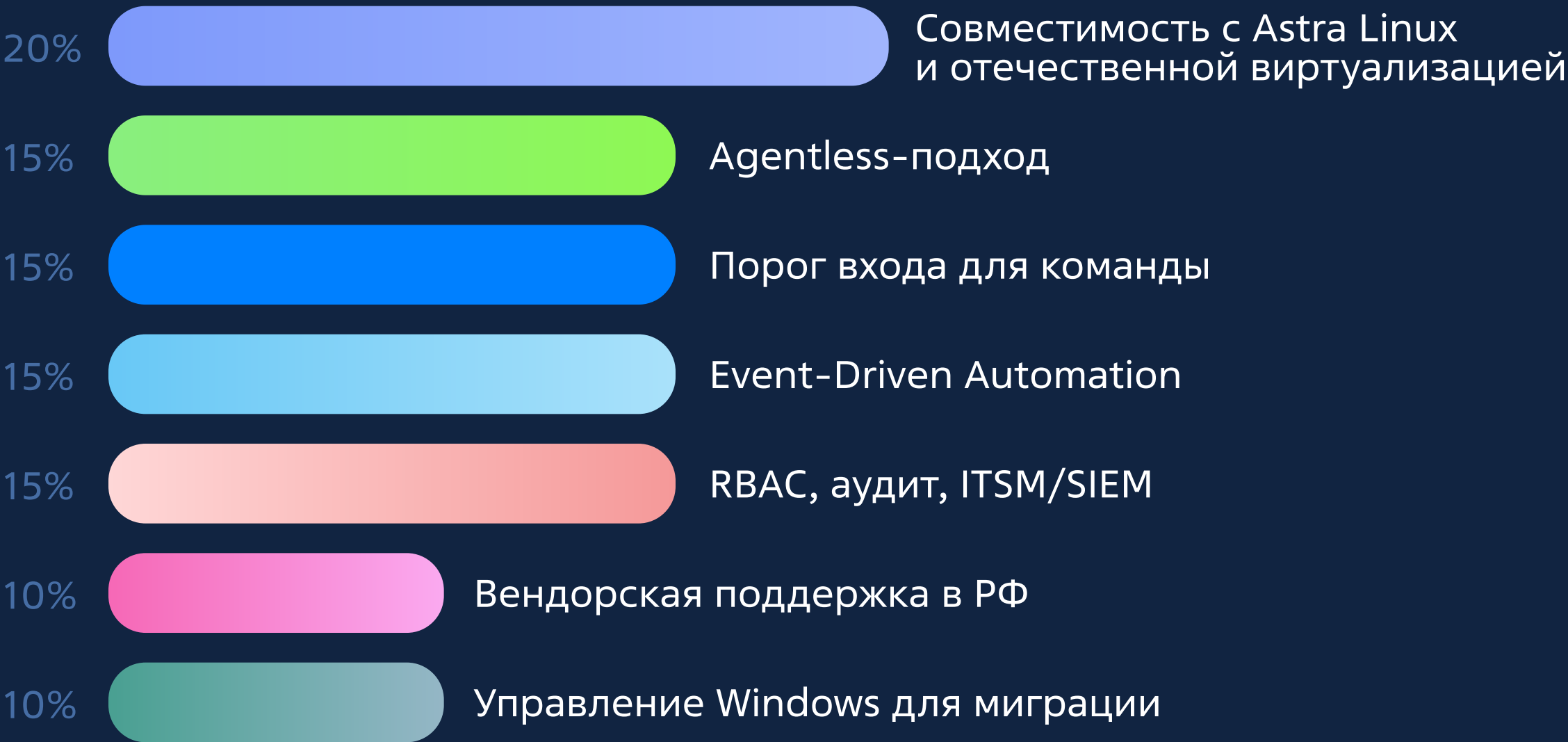
**Поэтому первым шагом стала не автоматизация,  
а стандартизация**

# Выбор технологии: управляемая модель



Q1-Q2 2024: критерии, веса и честное сравнение альтернатив

## 7 критериев оценки



## Краткое сравнение

|                      |                                 |                          |
|----------------------|---------------------------------|--------------------------|
| SaltStack            | real-time, 10k+ узлов           | агенты, сложнее с ИБ     |
| Puppet / Chef        | desired state, зрелость         | DSL, высокий порог входа |
| Terraform / OpenTofu | создание ресурса                | не конфигурация ОС       |
| AWX                  | agentless и знаком команде      | community без SLA/EDA    |
| ASTRA AUTOMATION     | Controller, Hub, EDA, поддержка | молодой продукт          |

## Вывод

Astra Automation подошла, потому что команда уже знала Ansible, нужны были agentless-подход, закрытый контур и вендорская ответственность

# Пилот проекта



Q2-Q3 2024: “живое” подразделение, 100 узлов и реальные ограничения

01 VM из шаблона + политики ИБ

02 Патч-менеджмент по расписанию

03 Восстановление из “золотого образа”

6 дней →  
40 мин

выделение VM  
через Controller

55% → 92%

compliance пилотной  
группы за 8 недель

8 ч → 3 ч

MTTR по восстановлению

## ЧТО ПОШЛО НЕ ТАК

Первые два месяца мы автоматизировали существующий хаос. Пришлось остановиться на 6 недель и утвердить эталонные шаблоны.

## ПОЧЕМУ ЭТО УСИЛИЛО ПРОЕКТ

Команда доказала не только технологию, но и способность остановиться, признать ограничение и изменить подход до масштабирования.

# Целевая модель: стандартизировать, контролировать, реагировать



Не «еще одна платформа», а контур управляемых изменений инфраструктуры

## ТРИ ЗАДАЧИ ТЕХНОЛОГИИ

### Стандартизировать

эталонные шаблоны  
и проверенный контент



### Контролировать

RBAC, аудит, запуск  
через Controller



### Реагировать

EDA: алерты →  
автоматическое действие



ITSM / каталог

заявка и право



Automation Controller

оркестрация, RBAC, расписание



Private Hub

проверенный контент  
в закрытом контуре



EDA

алерт →  
действие



Инфраструктура

Linux, Windows, сеть,  
виртуализация

## Ручной вход на сервер — исключение, требующее обоснования

# Путь к системной цели



Q4 2024 — 2026: от фундамента к self-healing и оптимизации



## Системная цель

**Любое изменение инфраструктуры проходит через код, проверенный контент и аудируемый запуск**

# Рост инфраструктуры больше не требует роста ручного труда



Результаты лучше всего показывают, что изменилась не скорость операции, а модель управления

18 минут

время выделения VM  
было: 6 дней

87%

соответствие политикам ИБ  
было: 55%

2,5 ч

MTTR АРМ  
было: 8 ч

-17%

затраты на лицензии  
и инфраструктуру

210 VM

выведено «забытых»  
ресурсов right-sizing  
+ очистка

-31%

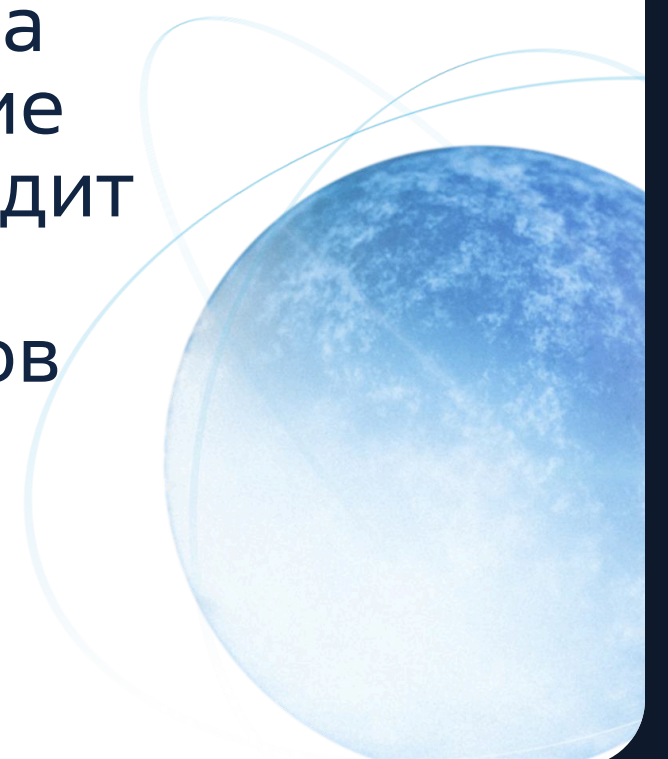
время  
администраторов  
на рутину  
было: 60%

≈11 FTE

перенаправлено  
с рутины  
на автоматизацию

## КОНТРОЛЬНЫЕ ЭФФЕКТЫ

0 инцидентов из-за дрейфа  
конфигураций за последние  
9 месяцев. Аудит ИБ проходит  
по отчетам Controller, а не  
по опросу администраторов



## Мы автоматизировали не действия администратора. Мы стандартизировали способ управления инфраструктурой

6 дней → 18 минут

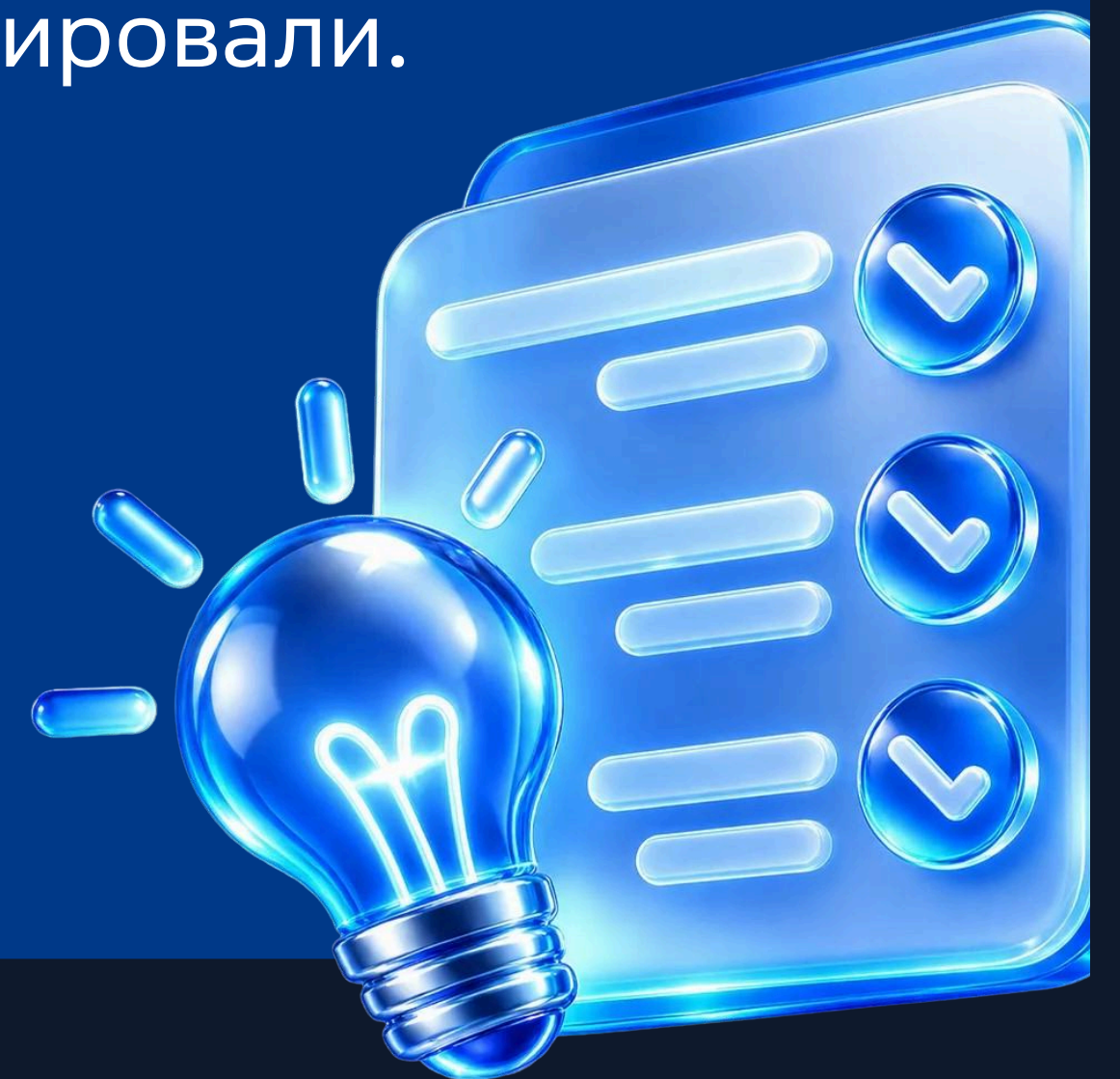
55% → 87% compliance

-31% рутины

### Наблюдения технократа

Самый сложный playbook оказался не про кластер, а про автоматическое закрытие заявок в ITSM. Инфраструктура сдалась быстрее бюрократии.

Спасибо. Вопросы принимаются вручную — этот процесс мы пока не автоматизировали.





Спасибо  
за внимание!