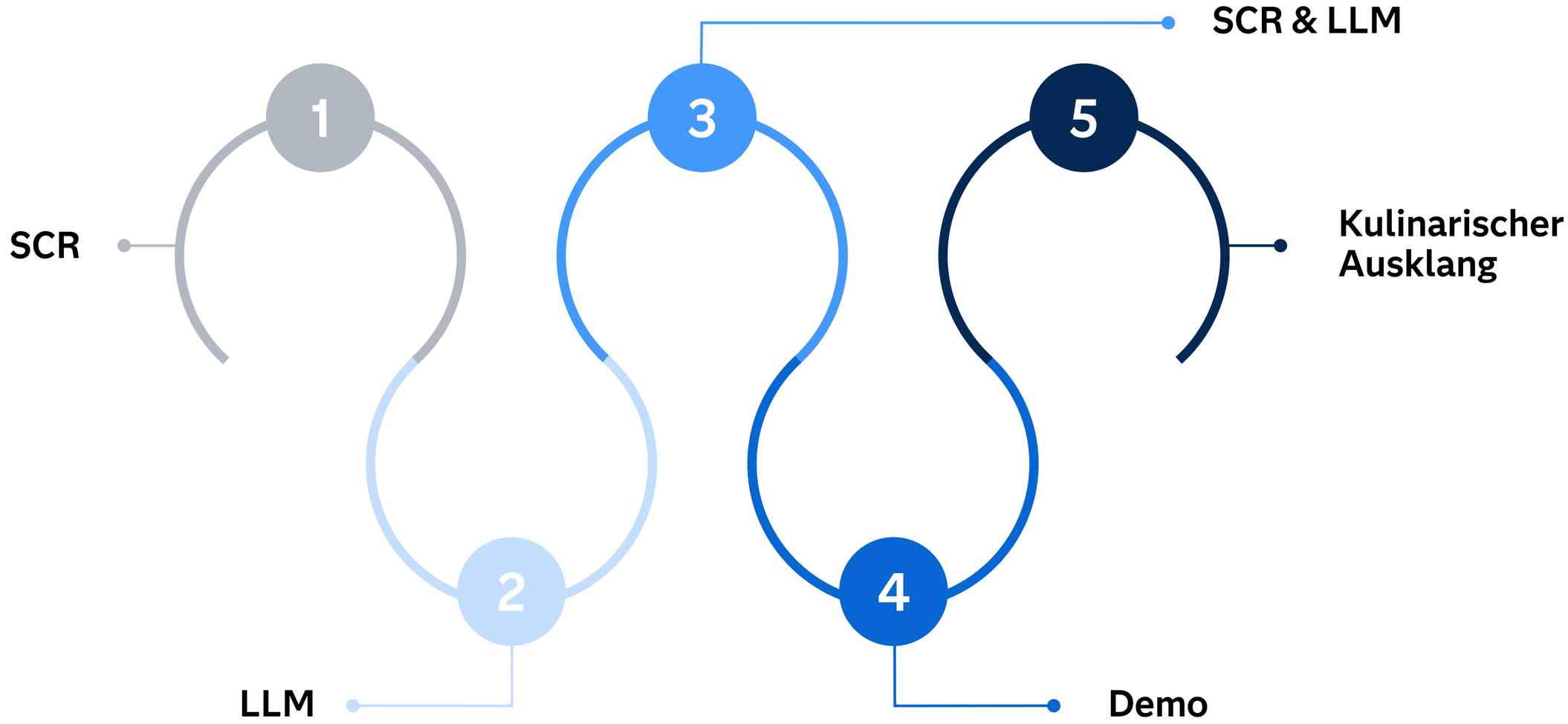
Large Language Models & SAS Container Runtime

Die perfekte Kombination!

David Weik, SAS R&D

Agenda



SAS Container Runtimer

Klingt wie Fußball, ist es aber nicht

Was ist die SAS Container Runtime?

SAS Container Runtime ist ein leichtgewichtiger, mit der Open Container Initiative (OCI) kompatibler Container, der SAS-Modelle und Entscheidungen auswertet. Damit können Modelle und Entscheidungen auf jedem OCI-kompatiblen Rechensystem ausführen, einschließlich Clustern, die auf Docker und Kubernetes basieren. Bereitstellungen in der Cloud oder auf lokalen Systemen werden unterstützt

Das Erstellen eines SAS Container Runtime Image, erfolgt über das Veröffentlichen von Modellen aus dem SAS Model Manager oder Entscheidungen aus SAS Intelligent Decisioning. Das Image kombiniert eine Basisschicht aus wichtigen Systemdateien, die von SAS bereitgestellt werden, mit Modell- oder Entscheidungslaufzeit-Codedateien

Sechs Eigenschaften von SCR

Minimaler Speicherplatzbedarf

Eigenständig

Unveränderliches Design

Schneller Start

Flexible Bereitstellungsoptionen

Hohe Skalierbarkeit

Arbeiten mit Python

Eine Funktion sie alle zu binden

- def scoreModel(INPUTS):
  “””Output: OUTPUTS”””
  <Hier bin ich König>
  return OUTPUTS
- Reines Open-Source Image in SCR
- Packages über ein requirements.json mitgeben

Mehr zu SCR in Hans-Joachim Ederts Vortrag (PNT > 2023 > 2023.09)

LLM

Sprachmodelle in einer Nussschale

Mehr zu LLMs und der Einordnung in die Analytik in Tamara Fischer und Annete Almers Vortrag (PNT > 2023 > 2023.09)

Lokale Modelle & Online Modelle

Von In-House bis zugekauft – welches Modell is am besten?

Rank* (UB)	Rank (StyleCtrl)	Model	Arena Score	95% CI	Votes	Organization	License	Knowledge Cutoff
1	1	ChatGPT-4o-latest (2024-09-03)	1339	+4/-4	28488	OpenAI	Proprietary	2023/10
1	1	o1-preview	1335	+4/-5	17562	OpenAI	Proprietary	2023/10
3	3	o1-mini	1313	+4/-4	17919	OpenAI	Proprietary	2023/10
3	3	Gemini-1.5-Pro-002	1305	+5/-4	11430	Google	Proprietary	Unknown
4	3	Gemini-1.5-Pro-Exp-0827	1299	+4/-3	32437	Google	Proprietary	2023/11
6	8	Grok-2-08-13	1291	+3/-3	35661	xAI	Proprietary	2024/3
6	9	Yi-Lightning	1287	+5/-3	13262	01 AI	Proprietary	Unknown
7	5	GPT-4o-2024-05-13	1285	+3/-2	99251	OpenAI	Proprietary	2023/10
9	15	GLM-4-Plus	1274	+5/-5	13674	Zhipu AI	Proprietary	Unknown
9	17	GPT-4o-mini-2024-07-18	1274	+4/-3	38831	OpenAI	Proprietary	2023/10
9	13	Gemini-1.5-Flash-Exp-0827	1269	+3/-4	25555	Google	Proprietary	2023/11
9	20	Gemini-1.5-Flash-002	1269	+8/-5	8957	Google	Proprietary	Unknown
9	5	Claude 3.5 Sonnet	1268	+3/-3	75957	Anthropic	Proprietary	2024/4
9	24	Grok-2-Mini-08-13	1267	+3/-5	30597	xAI	Proprietary	2024/3
9	7	Meta-Llama-3.1-405b-Instruct-bf16	1266	+5/-4	14496	Meta	Llama 3.1 Community	2023/12
10	8	Meta-Llama-3.1-405b-Instruct-fp8	1267	+3/-4	39428	Meta	Llama 3.1 Community	2023/12
10	7	Gemini Advanced App (2024-05-14)	1267	+3/-2	52235	Google	Proprietary	Online

https://lmarena.ai/ Leaderboard abgerufen am 22.10.2024

Viele Modelle, viele Möglichkeiten, viel Wildwuchs

- Welche Modelle können im Unternehmen genutzt werden?
- Wer nutzt welche Modelle und wie?
- Wie gehe ich mit unterschiedlichen APIs der Modelle um?
- Muss ich für jedes Modelle eine neue Integration schaffen?
- Wie können die Modelle in Geschäftsprozesse integriert werden?
- Wie sieht das Thema Kostenkontrolle aus?
- Wie können Abteilungen von einander lernen?
- Einer will programmierern, einer will es visuell nutzen, der nächste will es einfach als Teil seiner Anwendung, und so weiter und so weiter

SCR & LLM

Ah der Vortragstitel

Eine Standard API

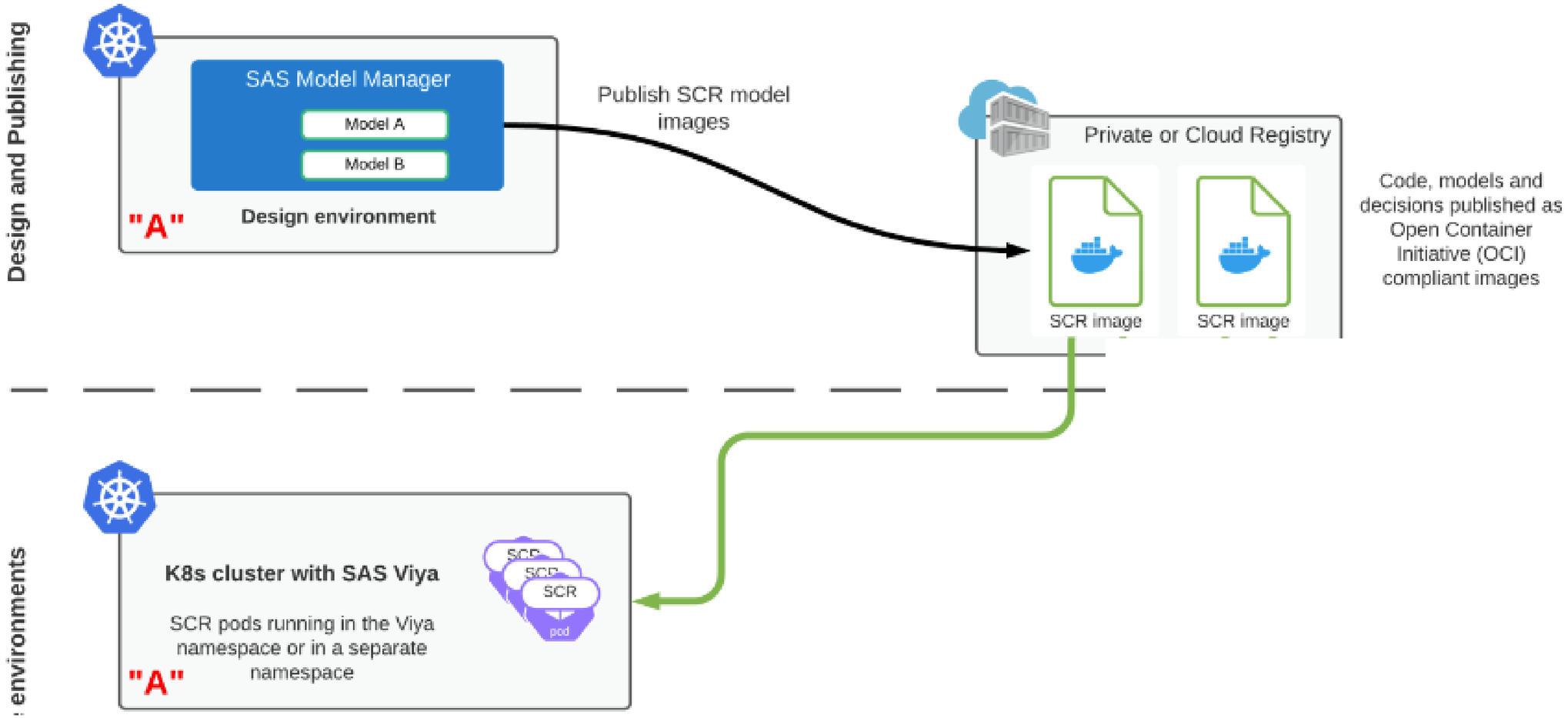
```
def scoreModel(userPrompt, systemPrompt, options):
    "Output: response, run_time, prompt_length, output_length“
    <LLM>
    return response, run_time, prompt_length, output_length
```

Alle Arten von Modellen

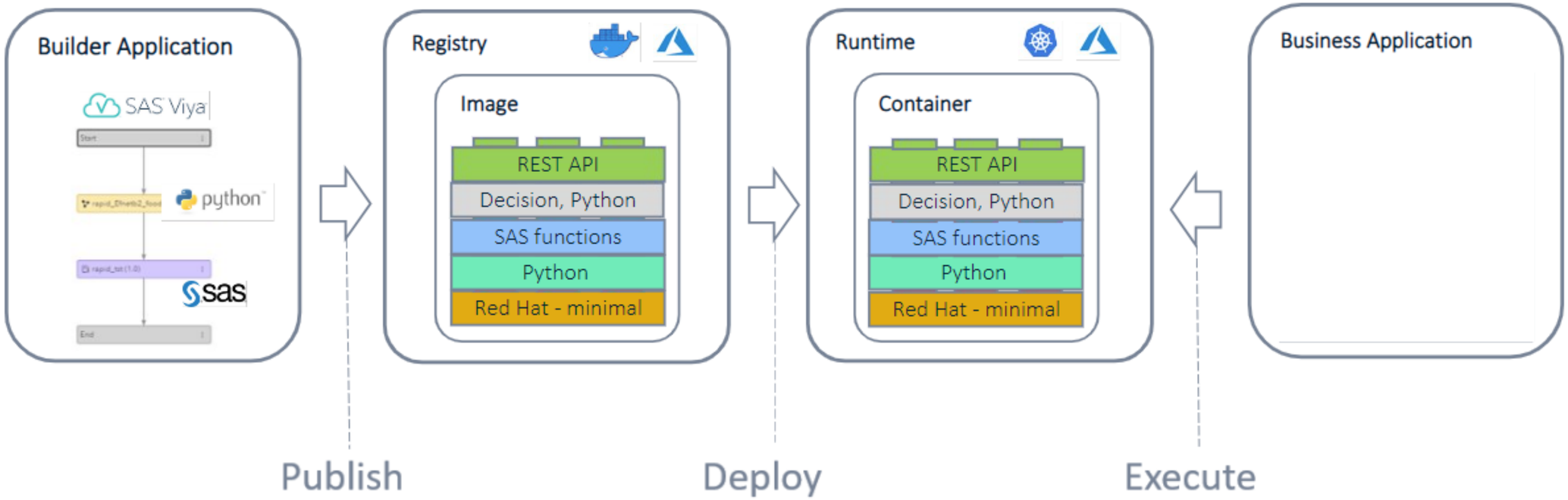
- claude_2_0
- claude_2_1
- claude_haiku_3
- claude_opus_3
- claude_sonnet_3
- claude_sonnet_3_5
- gpt_4o_2024_05_13
- gpt_4o_mini_2024_07_18
- mistral_nemo
- phi_3_mini_4k
- smollm_135m
- smollm_17b
- smollm_360m
- README.md
- baseScore.py
- inputVar.json
- modelConfiguration.json
- options.json
- outputVar.json
- requirements.json

Deployment per Azure DevOps

Chief Architect → Benjamin Walther!



Übersicht



STOP – DEMO TIME

Fragen? Feedback? Ideen?